Dependability Driven System Level Co-Design and Optimization of Embedded Systems

Vom Fachbereich Informatik der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades eines Doktor-Ingenieur (**Dr.-Ing.**)

vorgelegt von

Md. Shariful Islam

aus Bangladesch

Referenten:
Prof. Neeraj Suri, Ph.D.
Prof. András Pataricza, Ph.D.

Datum der Einreichung: 16.10.2008
Datum der mündlichen Prüfung: 10.12.2008

Darmstadt 2008
D17

Executive Summary

Embedded systems are becoming pervasive in diverse application domains such as automotive, avionic, medical, control and their functionality is increasingly defined by software (SW). Such systems especially in safety-critical (SC) applications, with implications on system *dependability* and *real-time* must be designed to be dependable (fault tolerant) enough and have to meet timing requirements in order to avoid any potential catastrophic consequences. More and more new and innovative functionality is being integrated into such systems, invariably leading to a heterogeneous environment consisting of applications of *mixed-criticality* (SC and non-SC), each with associated extra-functional requirements such as *dependability*, *timing*, *resources* and *power consumption*. Efficient system design methods and techniques are needed to be developed to integrate these diverse applications across limited hardware (HW) resources.

This thesis develops a novel dependability-driven system level SW-HW co-design methodology which systematically guides the design and optimization of such embedded systems from requirements analysis phase through integration to the prototyping. We first develop the concept of a consolidated mapping of SC and non-SC applications onto a common distributed computing architecture such that their operational delineation is maintained over the integration. We then devise an optimization based co-design approach through quantifying the various design objectives/variables. Our aim is to develop the design methodology for an integrated embedded architecture.

A heuristic based systematic mapping process is elaborated for integrating varied criticality applications. A set of functional and extra-functional *requirements* and *constraints* are satisfied during the mapping. At an early design stage, the mapping considers rigorous design strategies such as *fault tolerance, fault/error containment, robust partitioning, timeliness, resource* and *power consumption*. Dependability is ensured through replication of application jobs with high criticality and a schedulability analysis is presented for guaranteeing the timeliness properties. The developed mapping algorithm generates an initial feasible solution and guides the optimization in a unified and efficient way.

We develop a comprehensive multi variable optimization (MVO) framework which *quantifies* and *optimizes* a set of competing variables from dependability, real-time and resource perspectives. During the optimization process the satisfaction of constraints is maintained. The key aspect of the approach is to enhance dependability by using fault containment mechanisms including the quantification and estimation of the considered design variables.

The framework is extended by quantifying and modeling the reliability and system level power consumption as design variables.

In order to evaluate and validate the developed methods and techniques presented in the thesis, we have performed extensive experiments. Throughout the thesis we illustrate our ideas and concept using real-life automotive examples (where these techniques were actually validated). The concept is applied to a supporting tool set where we develop a prototype of the system level co-design approach. The prototype is created adhering to a *transformation based design* process.

Kurzfassung

Eingebettete Systeme sind in allen Bereichen der Automobil-, Avionik- und Kontrollanwendungen gegenwärtig und werden zunehmend durch Software (SW) definiert. Solche Systeme mit direkter Auswirkung auf die Zuverlässigkeit und Echtzeit müssen zuverlässig (fehlertolerant) genug sein und strenge Echtzeitanforderungen erfüllen um mögliche katastrophale Folgen wie den Verlust von Menschenleben, Schaden für die Umwelt oder den Verlust von Eigenschaften zu vermeiden. Die Integration von vielen neuen und innovativen Funktionalitten in solche Systeme führt unweigerlich zu heterogenen Umgebungen, bestehend aus sicherheitskritischen als auch nicht-sicherheitskritischen Anwendungen (mixed-criticality), jeweils mit den damit verbundenen extrafunktionalen Anforderungen bezüglich Zuverlässigkeit, Echtzeit und Ressourcen. Effiziente System-Entwurfsmethoden und Techniken sind erforderlich, solch unterschiedliche Anwendungen in begrenzten Hardware (HW) Ressourcen zu integrieren.

In dieser Dissertation entwickeln wir eine neue, verlässlichkeitsorientierte SW-HW Co-Design-Methode für die Systemebene, die den Entwicklungsprozess und die Optimierung solch eingebetteter Systeme von der Anforderungsanalyse über die Integrationsphase hin zur Realisierung des Prototypen in systematischer Weise führt. Zunächst entwickeln wir das Konzept einer gemeinsamen Abbildung von sicherheitskritischen als auch nicht-sicherheitskritischen Anwendungen auf eine gemeinsame, verteilte Rechnerarchitektur, so dass ihre operative Abgrenzung über die Integration erreicht wird. Weiterhin haben wir durch Quantifizierung der verschiedenen Entwurfsziele und Variablen einen optimierungsbasierten Co-Design-Ansatz erstellt. Unser Ziel ist es eine integrierte, eingebettete Architektur zu entwerfen.

Ein systematischer, auf Heuristiken basierter Abbildungsprozess wird für die Integration von Anwendungen unterschiedlicher Kritikalität entwickelt. Eine Reihe von funktionalen und extra-funktionalen Anforderungen und Randbedingungen sind während des Abbildungsprozesses erfüllt. Schon in der frühen Entwurfsphase berücksichtigt der Abbildungsprozess strenge Designstrategien wie Fehlertoleranz, Fehlereingrenzung, robuste Partitionierung, Echtzeit, Ressourcen- und Energieverbrauch. Zuverlässigkeit wird durch Replikation von Anwendungen mit hoher Kritikalität garantiert, die Echtzeiteigenschaften durch eine Schedulability-Analyse gewährleistet. Der entwickelte Abbildungsalgorithmus generiert zuerst eine zulässige Lösung und führt dann die Optimierung in einheitlicher und effizienter Weise durch.

Ein umfassendes Rahmenwerk für eine Multi-Variablen-Optimierung (MVO) wird entwickelt, das eine Reihe von konkurrierenden Variablen im Hinblick auf Zuverlässigkeit, Echtzeit und Ressourcenverbrauch quantifiziert und op-

timiert. Während des ganzen Optimierungsprozesses wird die Einhaltung der Randbedingungen garantiert. Der wichtigste Aspekt dieses Konzepts ist die Erhöhung der Zuverlässigkeit durch die Verwendung von Fehlereingrenzungsmechanismen wie Quantifizierung und Abschätzung der betrachteten Design-Variablen. Das Rahmenwerk wird dahingehend erweitert, dass die Zuverlässigkeit des Systems und der Stromverbrauch auf Systemebene als Design-Variablen quantifiziert und modelliert werden. Zur Evaluierung und Validierung der in dieser Dissertation entwickelten Methoden und Techniken werden umfangreiche Experimente durchgeführt. Im gesamten Verlauf der vorliegenden Arbeit, werden unsere Ideen und Konzepte anhand von praxisnahen Beispielen aus der Automobilentwicklung erläutert (wobei der Einsatz dieser Techniken zugleich validiert wird). Das Konzept wird auf eine Sammlung von Werkzeugen angewendet wobei wir einen Prototypen unseres Co-Design Ansatzes für die Systemebene entwickeln. Der Prototyp wird gemäß den Vorgaben eines transformationsbasierten Entwurfsprozesses erstellt.

Acknowledgements

With great pleasure, I would like to take this opportunity to thank all the people who have supported me in the successful completion of this thesis. At first I would like to express my sincere thanks to my advisor Prof. Neeraj Suri for his excellent supervision, scholastic guidance and constructive suggestions during this four years of working in the DEEDS group. A very special thanks to Prof. András Pataricza for accepting to be my external supervisor.

I am very thankful to all my past and present colleagues including Ute and Sabine in the DEEDS group for their support and for a nice working environment. Some of my research work has been done in collaboration with the EC DECOS project, thanks to several academic and industrial partners for fruitful discussion during the project work. I have got the opportunity to apply my developed concept in practice. I am also very grateful for the financial support from that project.

Last but not least I would like to express my thank to my parents, my wife and my siblings for their support, encouragement and love during this time period.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Embedded systems are becoming pervasive in our daily lives as well as in aspects where their proper functioning is crucial. Albeit hidden from the user, in reality we are surrounded by varied complexity embedded systems ranging from simple sensors, to wristwatch to cell phones, to the higher level *safety critical* systems used in *automotive, aerospace*, medical and control applications. Our growing use of embedded systems is driven by their sustained delivery of desired services despite the occurrence of perturbations. The perturbations range from electromechanical stresses, to system, software and communication level failures. For safety-critical systems, the need is to sustain operations in the presence of perturbations (*dependability*), this is often complemented by temporal requirements and the time duration by which a service is required (*real-time*) – also irrespective of the encountered perturbations.

The trend for the future is that more systems will contain computer-controlled/electronic components, i.e., will only increase the embedded controlled systems. Automotive original equipment manufacturer (OEM) such as DaimlerChrysler states that more than 90% of innovation (and hence value added) in a car will be in electronics and other OEM BMW indicates that more than 30% of the cost of manufacturing a car resides in the electronic components [1]. The worldwide value creation in automotive electrical and electronics systems, including SW, amounts to an estimated 127 billion EUR in 2002 and an expected 317 billion EUR in 2015 [2]. The trend is that use of electronics in modern car replacing critical mechanical and hydraulic components is increasing. The number and type of critical functions implemented by embedded electronic systems aboard a car and aeroplane is evolving as well. It is unforeseen that this trend will stop in near future. The design of such systems is becoming challenging and needs sound methods and techniques.

In general, *embedded systems* are defined as a class of special purpose computing system and are usually embedded (tightly or loosely) within a larger system; functionality is mostly fixed and dedicated; working very often in a reactive mode; responding frequently to external inputs; implemented by numerous concurrently working processes; infrequently reprogrammed; extensively sensitive concerning cost, power and performance criteria; have hard *dependability* [3] and correctness constraints, e.g., a brake-by-wire system aboard a car has to work bug-free and without interruption in any circumstances. The latter types are called as *dependable embedded systems*. These types of system may consist of micromechanical, microelectronic and increasingly use software components hence are heterogeneous containing hardware (HW) and software (SW) parts. Nearly all functions, for example, in vehicles and in aeroplane are controlled by *software components* (SW-Cs) running on microprocessors. The complexity of designing such systems is however growing at a very high pace through the integration of *mixed-criticality* applications (having both safety-critical and non-safety critical applications), through the use of multiprocessor cores, system-on-chip (SOC) architecture and through the high interactions between distributed applications. The constraints with respect to both functionality and *extra-functionality* are getting tighter as well. The extra-functional properties includes *dependability/fault tolerance (FT)*[1]*, reliability, timeliness, power, cost and time-to-market.*

In order to obtain a *dependable* operation of embedded systems while maintaining *performance/timeliness* and *resource efficiency*, the design process of such complex systems is becoming an important issue and also difficult at the same time to meet varied requirements. Though the design approach presented in this thesis is applicable for various embedded systems like automotive, avionics, aerospace, control, seaborne, embedded mobile and wireless systems, throughout this writeup we focus on examples for automotive applications, where we devote to apply our developed methods and techniques, for both *safety-critical (SC) and non-SC* specially for *X-by-Wire* (XBW) [4; 5; 6] and other critical applications where 'X' stands for any SC applications such as Brake-, Steer-, Flight-by-Wire. The critical applications include powertrain, car engine control, heart pace-makers, nuclear power plant control and military radar systems. We provide a brief description of different classes of automotive applications as follows.

Automotive Applications: A new class of computer-driven automotive applications is emerging known as *XBW* systems which are categorized as composite SC and *hard real-time* (RT) embedded applications. These types

[1]The terms dependability and FT will be used synonymously in the thesis.

of functionalities are used to control the movement of the car or assisting the driver. They are increasingly being implemented as SW replacing the mechanical and hydraulic systems in the car. The key benefits being enhanced processing capabilities, weight and cost. They often impose a high dependability requirements and tight RT constraints, e.g., the driver assistance system has to respond to the environment within a few milliseconds once it gets a request from the driver. The response has to be made correctly even in presence of any perturbations. These types of applications further include electronic stability programs, lane assistance, anti-lock brake systems, adaptive cruise control etc. Another type of applications implemented as SW-Cs is called as *body electronics*. These applications are less critical comparative to the ones defined earlier. The functionalities which control the simple electronic devices in the vehicle such as doors and window control, windshield wipers, lighting are categorized into this class. Usually they are classified in the category of safety relevant and also as non-SC applications. Recently many new and innovative functionalities are increasingly being deployed in the car for user comfort and communication. These are classified as *infotainment and telematics* applications. Examples of such applications include instrument cluster, in-car navigation systems, multimedia systems, car radio, hands-free phones, air conditioning system, car-to-car communication systems, etc. These are usually categorized as non-SC applications. During the design process care has to be taken such that there is no erroneous data or control flow from these types of applications or from applications of body electronics to the SC applications, i.e., the *fault/error propagation* from non-SC to SC application have to be prohibited by design.

Given the prime focus on designing SC embedded systems, we investigate some common idiosyncrasies of SC systems (as follows) which make the design of such systems challenging and look for new methodologies and tools.

- For embedded systems both the functional and extra-functional attributes are extensively defined by SW-C, i.e., by the so called embedded SW,
- These types of systems are composed of both SW-Cs and HW components (HW-Cs) which interact in order to perform the given task,
- Embedded system applications are distributed in nature and should be implemented over several microprocessors and HW architecture,
- The system often operates in SC contexts and has to provide services dependably, and

- Have to fulfill stringent *real-time* (RT) constraints, as well as constraints concerning timeliness, cost, weight, power and resource consumption.

The methodologies and techniques proposed in this thesis consider the conceptual and applied guidelines *(a)* to integrate such varied functionalities and characteristics of both SC and non-SC embedded applications, *(b)* to achieve dependability/FT, and *(c)* is followed by RT/timeliness, performance, power, resource and cost driven principles.

This chapter further provides a description of designing dependable embedded system at higher abstraction level, i.e., at *system level*. An introduction of different design techniques such as *federated and integrated* design approach, the need for *SW-HW co-design/integration* [7; 8; 9] and some evolving design challenges are described. The design at abstraction or system level implies the design process that hides the unnecessary implementation details. A short overview of system design for achieving and optimizing dependability is depicted. Subsequently the chapter describes the problem statement of the work and thesis research relevant questions. Finally the main contributions of the thesis are previewed including the publications and a short thesis outline is depicted.

1.1 System Level Design and Challenges

The design methods and techniques of embedded systems vary considerably with respect to the application domain under design and the type of controller and/or product under development. The applications are classified into two main categories of SC and non-SC with different requirements and characteristics. Many embedded controllers operate on systems that may cause severe damages to people and property if they malfunction, i.e., they are SC. These systems have to provide correct services at the correct time despite of any perturbations. The emergence of XBW technologies in the automotive, avionics and rail industry is significantly increasing the number and importance of such applications. The main advantages of using these technologies in comparison with conventional mechanical systems are improvements of enhanced functionality, response characteristics, dependability, weight, power etc.

The design choices and strategies of SC systems differ from the design of non-SC systems. SC systems must achieve a level of dependability which should be better than the dependability of any of its constituting SW and HW components. These systems must also have to meet stringent timing requirements in presence of *faults*, i.e., they are *hard-RT*. They need to be assured

for deterministic and predictable system behavior. For instance violation of the timing constraints can have catastrophic consequences. Whereas for the design of non-SC systems the emphasis lies on low cost, flexibility, resource efficiency and requires less stringent dependability and timing requirements. They are usually *soft-RT* and can run in a performance degradation mode as well. For instance in an engine control system not meeting the timing constraints can lead to a less efficient fuel consumption and more exhaust.

The consolidated design of these types of SC and non-SC embedded systems needs rigorous strategies and must be designed to be dependable (fault tolerant) enough and have to meet deadline requirement, i.e, the instant before which a result must be delivered, in order to avoid disastrous events such as loss of human lives, loss of properties or environmental harm. The probability of any fault/error propagation from non-SC to SC applications has to be prohibited by design. In the following, we describe and compare two design approaches practicing by automotive and avionics research community such as *federated* and *integrated* design approach.

The system level dependable design of embedded systems refers to the design of such systems at higher level of abstraction providing dependability by design and fulfilling different properties such as RT, power and resource consumption. The system level design is started specifying the system and constructing the HW and SW model separately out of the defined specification. Then the SW-Cs are mapped onto HW-Cs. Mapping is decomposed into two subproblems: *allocation and scheduling*. Figure 1.1 depicts the abstract view of essential steps of such design developed in this thesis. The design

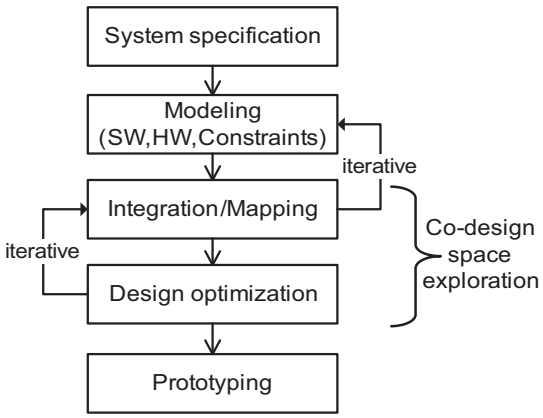

Figure 1.1: Abstract view of system level design steps

flow includes activities such as system specification, SW, HW and constraints modeling, dependability and RT driven *SW-HW integration/mapping*, design optimization and the prototype of the system level design. Some design steps

are performed iteratively and refined by *iterative improvement*, e.g., mapping and its optimization. The design/co-design space exploration enables to explore and exploit the search spaces by investigating alterative mappings and their optimization in order to be able to find an optimized mapping from the global design space. All these steps are detailed throughout different chapters of the thesis.

Federated Design Approach

The traditional federated design paradigm implements a function onto a single node through the vertical integration as shown in Figure 1.2. In automotive community node is usually known as *electrical control unit* or ECU. Figure 1.2 shows the integration of three different SC (SC1, SC2 and SC3) and two different non-SC applications (nSC1 and nSC2). Applications are further decomposed into executable fragments referred as jobs (Job11, Job12, Job41, ..., Job3m, Job5p). Jobs have basic communication capabilities for exchanging information with other jobs. For instance Job11, Job12 and Job13 communicate with each other to provide the necessary service of SC1.

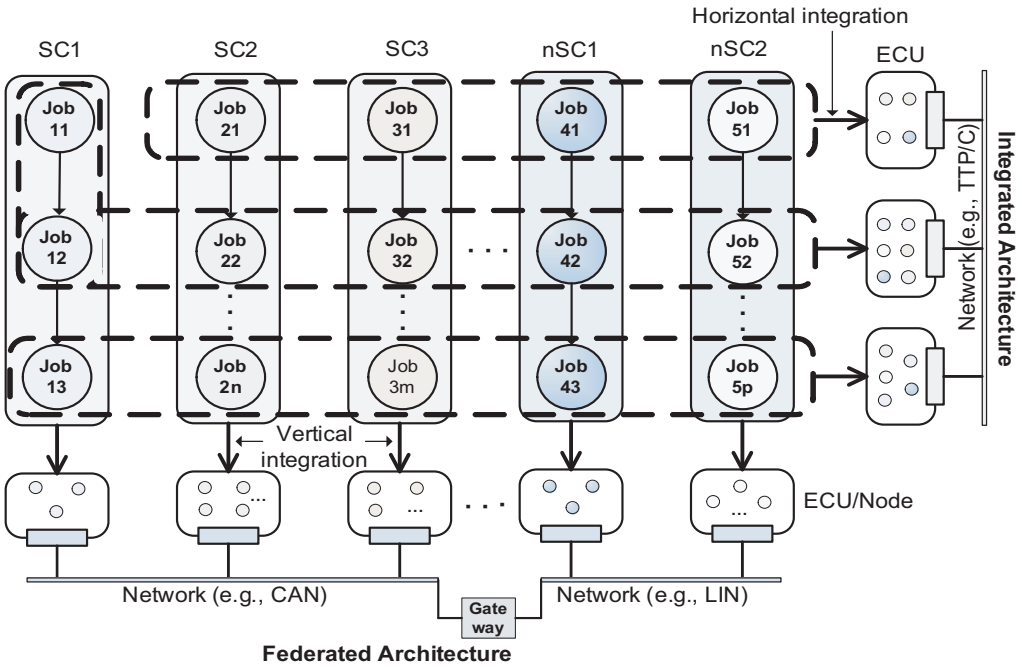

Figure 1.2: Federated and integrated design approach

In the federated or the customized solution approach, FT is provided by simply replicating each of the ECU therefore the number of ECUs be-

come double/triple for each SC application. Due to its 1 function to 1 node mapping philosophy, adding a new functionality in the system tantamount a new node. However this approach provides spatial partitioning among the applications which facilitates *fault encapsulation and isolation* (fault in one application does not affect other application). Thus this type of design is though desirable for dependability prohibitively expensive from the number of ECUs, size, weight and cost perspectives. Moreover, as this design approach follows a customized implementation of 1 function to 1 node strategy, functions distribution is not possible, i.e., transferability of functions across nodes cannot be performed. For instance jobs of SC1 cannot be integrated with jobs of SC2 even there are enough resources available on the node where jobs of SC2 are assigned. Hence optimization cannot be applied for such a design approach resulting a practically in-efficient use of available resources. In the automotive domain, massive deployment of this design concept has led to a large number of ECU/nodes, sensors, actuators and networks aboard a car. A current premium car, for instance, implements about 270 functions and deploys over sixty to hundred nodes and five to twelve different communication networks [10; 11]. The networks may or may not be connected with each other via gateways. Gateways are used to establish a minimum exchange of information between networks as well as between different applications. The large number of nodes and networks have some undesirable consequences like high number of wires contact and high costs. The wiring and contact is one of the major causes of failure in automotive systems. Field data has shown that more than 30% of electrical failures are due to connector problems [12]. Design improvement is needed from this perspective as well.

The notable example of networks which are widely used for automotive systems includes controller area network (CAN) [13], local interconnect network (LIN) [14], media oriented system transport (MOST) [15], time-triggered protocol (TTP/C) [16], FlexRay [17] and time-triggered ethernet (TT Ethernet) [18]. TTP/C is a purely time-triggered network provides deterministic messages transmission for FT-RT systems which is a suitable candidate for the integrated architecture.

Integrated Design Approach

In order to tackle the challenges and drawbacks faced with the federated approach an alternative design paradigm has been introduced called as the integrated design approach. This approach promises cost saving through the reduction of complete resource/ECU replication. Fundamentally, a single node processor now executes a variety of both SC and non-SC applications. In this approach integration of mixed criticality applications onto a com-

mon distributed computing architecture is carried out as shown in Figure 1.2 marked with the horizontal integration. We see that jobs from different criticality applications can be assigned onto the same node. Thus this design approach allows the interplay among application jobs enabling efficient use of resources. FT is provided by active SW based replication where replicated functions are integrated onto different ECUs without adding new ECU/node in the architecture. As this approach requires less number of nodes and networks onboard a car, it improves dependability in terms of wiring and connectors [12]. A low number of nodes benefits the design complexity, wiring, mounting, HW cost, weight, space and many others. In an integrated architecture design concept, jobs from different SC applications as well as from non-SC applications are implemented on the same node hence integration of mixed criticality applications are carried out. *Influence* of non-SC to SC applications has to be prohibited by design. Influence is the probability of error propagation between modules where a module can be an application, a job, a processor core or a node.

Recently a number of customized design efforts have been made and the importance and benefits of such integrated approach is evident from the design concepts in avionics industry, e.g, the Integrated Modular Avionics (IMA) [19; 20; 21] as well as such design concept is currently being introduced in the automotive industry such as in AUTomotive Open System ARchitecture (AUTOSAR) [22] and in [12; 23]. Thus in order to *(a)* reduce development, production and maintenance cost, *(b)* increase the dependability of embedded applications, and *(c)* perform system optimization in the application domains of dependable RT embedded systems, it is necessary to develop the enabling technology to move from a federated to an integrated design approach. The design of such system needs support of some architectural and core services. An example of such an architecture is the Time-Triggered Architecture (TTA) [24] which is a distributed integrated architecture supports developing and implementing varied critical applications to the highest criticality class. TTA provides a set of core and high level services in order to support application jobs execution and ensures the predictability and deterministic behavior of SC applications. The services include deterministic and predictable message transmission, fault tolerant clock synchronization, strong fault isolation and consistent diagnosis of failing nodes.

The current approaches are predominantly customized designs, which are usually driven either by discrete FT or RT basis. The challenge remains to provide *composite* FT and RT design for an integrated embedded architecture (IEA) to achieve a certain level of dependability (FT and fault/error containment) including fault encapsulation and isolation as in federated architecture while meeting the RT, performance and power requirements and

constraints.

Design Challenges

As mentioned the complexity of embedded systems design is growing rapidly. Furthermore in one hand SC applications have introduced a new design dimension due to the distributed nature of the system. On the other hand one application may have control and data dependencies over other applications, e.g., collision avoidance system simultaneously need to control both the brake-by-wire and steer-by-wire systems [25]. These phenomena entail additional complexities, yet potentials for optimizations. The optimization features include the reduction of the number of needed ECUs, fewer mechanical parts, better performance, upgrading with new functionalities including safety aspects. The trend is that the number and complexity of functions as well as new innovative functions will increase drastically. Addition of new functionalities and increased complexities are making the present design methodologies rapidly obsolete. Furthermore the design has to ensure that the SC functionalities are not affected by the non-SC ones. [26] states that the development of methodologies, techniques and tools for system level design is the only solution to cope with the increasing complexity of embedded systems and the productivity gap. The design of such complex and interacted system is not possible by the existing traditional approaches, e.g., by the federated approach. Therefore the research presented in this thesis deals with the design issues for an integrated architecture at system level of abstraction. System level co-design is a new paradigm which focuses the embedded system design challenges at higher level of abstraction hiding the architecture specific implementation details as much as possible during different design steps.

In order to ease the design complexity, integrated system design should come up with guidelines, methodologies and tools [27] and needs a stepwise design process. The development of such design concept also calls for new forms of abstraction and design methodologies for bridging applications and the architecture components. Particularly the design methodologies and techniques should cope and meet with the following challenges:

- Growing complexity of embedded systems through adding new functionalities,

- Achieving a certain level of dependability/FT as well as meeting the other constraints such as RT, resources and power,

- Mapping of SW-Cs onto HW resources maintaining operational delineation of SC and non-SC applications,
- Transferability of functions, i.e., optimal interplay across functions integration and quantification of multiple design criteria and their optimization and trade-off analysis, and
- Reducing the development, production and manufacturing costs and shorten the time-to- and time-in-market.

Therefore there is a need for a comprehensive and composite (FT and RT) integration approach to cope with these challenges. The solution is the system level *SW-HW Co-design/Integration* where dependability and RT are the driving factors. *Co-design* addresses the concurrent design of SW and HW and their mutual relationship during the design process. A unified design methodology supporting design steps of system specification, HW and SW partitioning, integration, validation, prototyping is the overall goal of the co-design. Performing co-design at system level allows to efficiently explore large design space and more design possibilities at early design stages. Consequently, in this thesis we develop methodologies and techniques for such system level co-design keeping the main focus on design for dependability and its optimization.

1.2 Dependability by Design and Optimization

Design of embedded systems with implications on dependability has to ensure that each application produces correct output and preserves the safety of the operations in the presence of faults so that any potential catastrophic consequences can be avoided. The term dependability of a computing system is defined as the ability to deliver services that can justifiably be trusted [3]. To facilitate the conceptual understanding of dependability related terms used in this thesis, we first explain faults, errors and failures. We further describe their effects on system followed by a discussion on fault and error containment. Next, we explain how dependability can be achieved and enhanced.

Faults are the causes of failures in the system by being activated (becoming errors) and then propagating to the output of the system and there causing a failure. Figure 1.3 illustrates how a fault propagating to a error/failure of a source module (module A) can be the input (fault) of another module

(target module B) and so on [29]. Further more on these terms and notions can be found in [3; 28].

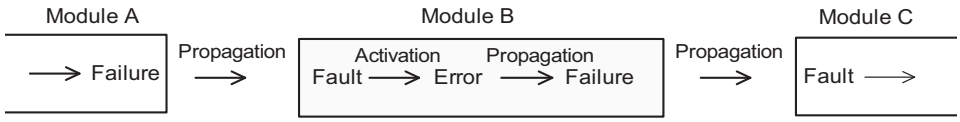

Figure 1.3: The fault, error and failure propagation

Faults, Errors and Failures

Faults, errors and failures are the reasons for a system to cease to perform its correct operations, i.e., they are the threats to dependability. The problem related to a fault is occurred at physical level, for an error it occurs on computational level and for a failure the problem occurs on a system level. During the operation of computing systems, events such as noise, bugs, broken connectors or wires may occur which can threaten the system ability to deliver the correct services are defined as *faults*. Different types of faults can affect the system which are development faults that occur during the development, physical faults which affect HW components and interaction faults that are due to the environmental disturbances or due to other external factors. A fault in itself may not threaten the proper functioning of the system but it has to be activated. When a fault is activated, an *error* is said to exist in that system. During the computation of a system if a faulty value is used then an error is said to propagate, i.e., there is error propagation. Due to the propagating error to the output of a system, the behavior of the system is deviated from what is prescribed by the specification, i.e., a *failure* is said to happen. Hence faults are reasons for errors and errors are reasons for failures. As shown in Figure 1.3, an error and a failure may themselves cause a fault/error/failure in another module lead the thinking of prohibition of propagation or containment of them within the same module.

Fault and Error Containment

If a fault/error is present in a module, it is possible for this fault/error to propagate to other modules and can cause multiple failures. The failure of a module due to the failure of another module is called a *cascading failure* which may risk generating the total system failure. In ultra-dependable systems [30] even a very small correlation of failures of the replicated units can have a significant impact on the overall dependability [31]. Fault and error containment is the process of isolating a fault and preventing propagation

of the effect of that fault, i.e., preventing error propagation throughout the system. The purpose is to limit the spread of the effect of faults, i.e., error from one module to another module of the same system or a different system. In this thesis we use the term *influence* in regard to the fault and error containment which is the inverse of these terms and defined as the probability of error propagation between modules.

Achieving Dependability

A dependable embedded system should be able to handle faults in individual SW and HW components, faults introduced by the designer during development, power failures or any other kinds of external perturbations or unexpected consequences and should still satisfy the specification. For designing such systems, there are several methods and techniques that can provide dependability, namely fault tolerance, fault prevention, fault removal and fault forecasting.

Fault tolerance (FT) technique aims to avoid service failures or to assure system correct functionality in presence of faults, i.e., to tolerate the effects of faults hence the name fault tolerance. FT is mainly achieved using some form of redundancy in spatial or in time domain. The redundancy allows either to mask or to detect a fault, with the location, containment and *recovery*. *Fault prevention* aims to prevent the occurrences or introduction of faults. The system is designed in such a way that fault does not occur which is termed as the fault prevention or avoidance. It is achieved by careful validation, efficient test and quality control techniques during specification, integration, implementation and fabrication stages of the design process. For example, a rigorous design review may eliminate many of the specification faults. *Fault removal* technique aims to reduce the number and severity of faults present in a system. It is usually performed during the development phase, e.g., by verification, diagnosis and correction as well as during the operational life of the system, e.g., by corrective and preventive maintenance. *Fault forecasting* technique aims to estimate the number of faults present in the system, possible future occurrences of faults, and the likely consequences of faults. It is performed by qualitative and quantitative evaluation of the system behavior with respect to the fault occurrences or activation.

We consider FT mechanism to achieve a certain level of dependability. While providing FT, other properties such as RT, resource consumption are satisfied as well. Embedded system designers particularly in the automotive sectors are emphasizing now on a strategic shift from simply achieving *feasibility*, to achieving *optimality*. A key aspect of this thesis is to enhance/optimize dependability through providing the fault/error containment

mechanisms such that the errors are not propagated but contained within a module, i.e., by reducing the influence. Overall, we want to assure that a system will perform as expected, i.e., a dependable embedded system is achieved by design.

1.3 Problem Statement and Thesis Research Questions

The constant growth of embedded systems design complexity due to integration of more and more critical, new and innovative functionalities into such systems, invariably leads to a heterogeneous environment consisting of applications of mixed criticality, each with associated dependability and RT requirements. Each application introduces design constraints such as SW complexity, cost, space, weight, resources, power and multiple other realization constraints making the overall system composition a complex resource allocation and optimization task. Thus efficient system design strategies are needed to integrate these diverse applications across limited HW resources while considering the interplay of FT and RT objectives including many others. The system level co-design and optimization is the solution for designing such integrated embedded architecture.

Traditional design techniques such as federated approach [32] are increasingly becoming limited for developing such systems. *Extra-functional* properties such as timeliness, FT and safety are introduced often late in the development process when the design is difficult and costly to change/upgrade. For example, FT is treated as an add-on requirement in the design process. A typical (and costly) approach being replicating the implementation, i.e., a so called federated approach. Investigations show that this approach fails to produce cost-effective dependable systems [12; 26; 33]. On the other hand embedded products have become increasingly complex and must be developed quickly that current design methodologies are no longer efficient [40]. Therefore integrated approaches are often advocated and recently been attracted by automotive and avionic industries where integration/mapping of different criticality applications onto a common distributed computing architecture is carried out. The mapping needs to satisfy a set of functional and extra-functional constraints and needs to perform in an efficient and optimized way. Furthermore the design complexity can be reduced significantly if the design and optimization process is carried out at system level [26; 41].

Mapping of mixed criticality and responsiveness[2] applications onto distributed shared resources is the most important and crucial step for system level design concept. A mapping is defined as: *(I)* assignment of jobs onto suitable HW nodes such that platform resource constraints and dependability requirements are met (*resource allocation*) and *(II)* ordering job executions in time (*scheduling*). In general there are K^Q ways of allocating Q jobs onto K nodes. Finding a feasible solution out of K^Q ways and meeting all the constraints for this particular problem is often NP hard [34] to solve in a tractable manner where a solution can be found in polynomial time [35]. Thus in general it requires development of heuristic techniques to solve those problems. On the other hand, existing approaches, e.g., [20; 36; 37; 38] usually do not address *(I)* and *(II)* together. Mostly scheduling is performed assuming a predetermined manual allocation, e.g., [20; 36]. This may not be possible for a fast growing embedded system where new functionalities and complexities (due to large number of design requirements and constraints) are increasing day-by-day. Thus intuitive mapping decisions are inherently limited beyond a given complexity. Consequently, we have developed a heuristics based *systematic resource allocation* approach for the consolidated mapping of SC and non-SC applications. Dependability/FT and RT requirements are the prime drivers for the proposed mapping and they are taken into consideration in the first step *(I)* of the mapping, i.e, extra-functionalities are considered early in the design process.

The design process further faces new challenges under various resource constraints and needs careful attention such that FT and RT requirements are not compromised but achieve to a certain level. For instance a commonly used value for dependability is 1 failure in 10^9 hours (1FIT), i.e., developing the system with ultrahigh *reliability* requirements [30]. Examples of resource constraints include limited physical resources, sharing the same distributed computing architecture and the pressure from automotive industry to use a less number of nodes. This phenomena entails the situation of considering optimization in the design process in order to develop as good solution as possible. The design optimization involves simultaneous consideration of several incompatible and often conflicting objectives. The complexity of the design endeavor becomes apparent when considering the huge design space of possible solutions on one hand, and many competing design objectives on the other [39]. Overall, better design methodologies are needed to handle such complex optimization problem. In order to tackle this, we propose a generic optimization framework considering different design *variables* from

[2]Responsiveness is a RT property where application has to respond within a certain time.

Dependability/FT, RT, power and resource consumption perspectives. A single variable refers to optimization of a single objective[3]. The approach is called as *Multi Variable Optimization (MVO)*, which takes into account the satisfaction of various constraints as well as the optimization of multiple competing variables that evaluate the optimality of a solution. The key challenge in optimization relies on modeling the estimation and quantification of the considered design variables.

Based on these premises we formulate the design problem with various thesis research questions. These are elaborated over different thesis chapters.

Thesis Research Questions

We pose the following research questions and briefly explain on each. The details are developed over the thesis writeup. We also mark the references of different thesis chapters and sections relating with the research contributions of the thesis.

Research Question 1 [RQ1]: *Why is a system level SW-HW Co-Design/Integration needed? Why do we need to shift from a federated to an integrated approach? How is the dependability driven co-design performed?*

Co-design addresses cooperative and interdisciplinary design of both HW and SW building a mutual relationship between both types of components as well as allows better trade-offs between them. The overall objective of doing co-design in this thesis is to develop *a unified design methodology* supporting various design steps of system specification, SW-HW mapping, optimization, analysis, evaluation and prototyping. If the design process is focused at the low level then a very limited set of trade-offs, optimizations and design refinements can be done since most of the crucial function and architecture decisions would have already been made. Thus the importance of doing co-design and doing at system level lies on several benefits such as managing the highest level of design complexity, performing the design not depending on the target architecture, performing proper optimization, shorten the time-to-market, reducing design efforts and reducing costs of designed products. In order to develop the enabling technology which *(a)* reduces SW and HW development, production and maintenance cost, *(b)* increases the dependability of embedded applications, and *(c)* performs system optimization, recently automotive and control industries have given attention for designing an integrated architecture versus the existing federated one.

A comprehensive dependability driven framework for system level co-design presented in Chapter 3 addresses RQ1.

[3]In this thesis objective and variable are used synonymously.

Research Question 2 [RQ2]: *What are the design constraints? How are they satisfied? How they can be handled efficiently? What are the design criteria?*

Constraints define the conditions that limit the possible designs from a dependability, RT, power or resource perspective. They are satisfied during the mapping process specifically during the allocation of jobs onto available HW resources. A list of functional and extra-functional constraints are given in Section 2.3 of Chapter 2. The constraints are satisfied in a prioritized manner. The constraints handling and satisfaction techniques are addressed in Chapter 3 and 4. Since we consider the design of dependable RT embedded systems, this thesis focuses on the criteria of dependability, RT, resource utilization and power including the satisfaction of constraints. Various design criteria are detailed in Chapter 3.

Research Question 3 [RQ3]: *How is the SW-HW mapping performed? How do we create a feasible mapping? Why is composite FT+RT the main focus? How is FT+RT achieved, i.e., how a certain level of FT and RT be reached? How can we prohibit the influence of non-SC on SC applications?*

A heuristic based systematic mapping process is developed for allocating SW-Cs onto HW nodes such that dependability and RT requirements are not compromised. For creating a feasible mapping, a set of constraints need to be satisfied as mentioned in the previous research question [RQ2]. Our target applications consider SC systems and since dependability is a primary design objective for such systems, the developed *system level co-design optimization* framework presented in Chapter 3 puts emphasis on FT through replication of highly critical application jobs and satisfies RT properties through *schedulability* analysis to provide the correctness of the function execution in time. The design strategies such as achieving dependability, reducing influence, providing schedulability and the SW-HW mapping process itself are described in Chapter 4. The *influence* reduction mechanisms (presented in Section 4.1.2 and 5.2.1) aim to prohibit the error propagation from non-SC to SC applications and then the system level partitioning mechanisms presented in Chapter 2.3 restrict the erroneous flow of information.

Research Question 4 [RQ4]: *What is the optimization of an integrated design? How is the multi variable optimization (MVO) carried out? How is the global search space guided? How are the variables of dependable embedded real-time systems estimated and quantified? How do we optimize different variables?*

Optimization of a design process is defined as the technique to find as good solution as possible. The process usually employs an iterative improvement

algorithm. The proposed *MVO* approach presented in Chapter 5 simultaneously quantifies and optimizes a set of competing design variables while maintaining the feasibility of the design. During the optimization we employ a *transformation operator* Γ in order to explore the co-design space. By employing the heuristics and constraints satisfaction techniques, the search process is initially guided to create a feasible solution which is then guide the optimization in a global design space. The MVO approach enhances dependability through quantifying and confining the influence, i.e., by reducing the error propagation probability between modules. A formulation of the optimization process and the selection, *quantification* and optimization of multiple variables for designing an integrated embedded architecture for FT-RT systems are addressed in Chapter 5. The process is then applied to an existing metaheuristic based optimization algorithm. The chapter further elaborates on [RQ4].

Research Question 5 [RQ5]: *How can we evaluate the performance and effectiveness of a co-design methodology? How do we know that the design is valid and optimal per se?*

We provide an empirical evaluation of the overall system design approach through performing extensive experiments. First, the initial feasible mapping generated by the heuristic approach is *validated* using an independent tool [42]. We *compare* the heuristics based approach with existing base line approaches. The experimental results show the effectiveness, performance and robustness of the mapping heuristic. We discuss more on this in Chapter 6. The experimental evaluation of the MVO approach shows the *convergency* to a global minima (a *near-optimal* solution would suffice) which gives a proof of optimality. Results show significant improvements (*quantitative gain*) of the considered design variables over the contemporary analytical initial feasibility solutions. The validation and experimental results are presented in Chapter 6 which elaborates [RQ5].

Research Question 6 [RQ6]: *Which techniques can reduce the design effort and shorten the time-to-market?*

We believe that designing embedded systems at system level and using the developed *prototype* in Chapter 7 will significantly reduce the design efforts and time.

1.4 Thesis Contributions

Overall in this thesis, we make the following contributions to the embedded system design and optimization research community.

Design Methodology: We develop a novel generic dependability driven co-design methodology which meticulously guides the system level design and optimization of integrated FT-RT embedded systems. [Chapter 3]

Rigorous Design Strategies: Rigorous design criteria such as classification of requirements and constraints, robust partitioning between mixed-criticality applications, SW reusability, FT, fault/error containment, timeliness, resources and power consumption are comprehensively addressed in our approach. The extra-functional requirements like dependability/FT, RT are taken into account at early design phase. Moreover we present potential design criteria of dependable embedded systems including essential co-design issues for performing the mapping and optimization. [Chapter 2, Chapter 3]

Consolidated Mapping: Based on heuristics a systematic mapping of SC and non-SC applications onto a common distributed computing platform is carried out such that their operational delineation is maintained over the integration. The design strategies stated above are the main drivers behind the mapping. The constraints satisfaction technique such as constraints prioritization, consistency enforcing is a viable strategy for creating the feasible mapping. This way the developed mapping algorithm generates an initial feasible solution and guides the design optimization in an efficient way. [Chapter 4]

Achieving and Enhancing Dependability: Dependability/FT is ensured through replication of high criticality application jobs. Jobs are replicated according to their degree of criticality and are disseminated over distinct nodes. We then enhance dependability by providing techniques to constrain the propagation of errors hence fault/error are not propagated but contained within a single node. While maintaining and improving the dependability we present a schedulability analysis for guaranteeing the timeliness/responsiveness properties. [Chapter 4, Chapter 5]

Quantification and Optimization: We develop an MVO approach which provides the quantification and optimization of different competing and conflicting design variables. The variables include *influences, scheduling length, and bandwidth utilization* which enable us to solve the MVO problem. The estimation and modeling of each of the design optimization variables is detailed in the thesis. We employ an existing optimization algorithm within the approach that enables a quantitative evaluation. [Chapter 5]

We extend the approach by quantifying and measuring the reliability and system level power consumption as optimization variables while considering a heterogeneous architecture comprises nodes of different speeds and failure rates. Overall the MVO approach develops a comprehensive and unified

framework for FT and RT driven integration. [Chapter 8]

Performance Evaluation: In order to evaluate and validate the approach, we perform extensive experiments which show the effectiveness (quality of the solution), performance (reducing the search space and finding a quick feasible solution) and robustness (consistent to perform the same mapping over many runs) of our design process. In order to proof the schedulability a validation of the allocation process is performed as well. Throughout different sections of the thesis we illustrate our ideas and concepts using different examples of automotive applications. An empirical evaluation including convergence and effectiveness of the MVO approach is provided. For a representative target study, our evaluation shows significant design improvements for the considered variables. [Chapter 6]

Prototyping: We have devised a prototype of system level co-design which includes a supporting tool set and technologies. For the prototyping, a transformation based design approach has been developed adhering to model driven development and platform based design principles. We believe that using our methodology and prototype the design time can be significantly reduced. [Chapter 7]

Papers Published During this Work

The following list of papers were published in scientific conferences and journals during the course of this thesis work.

- **Shariful Islam**, Neeraj Suri, András Balogh, György Csertán & András Pataricza, *A Transformation Based Design for Integrated Dependable Real-Time Embedded Systems*, Submitted to the Journal of Design Automation for Embedded Systems (DAES) (In review), 2008.

- **Shariful Islam** & Neeraj Suri, *A Multi Variable Optimization Approach for the Design of Integrated Dependable Real-Time Embedded Systems*, In the IFIP International Conference on Embedded and Ubiquitous Computing (EUC), 2007.

- **Shariful Islam**, Robert Lindström & Neeraj Suri, *Dependability Driven Integration of Mixed Criticality SW Components*, In the 9th IEEE International Symposium on Object and Component-oriented Real-time distributed Computing (ISORC), 2006.

- **Shariful Islam**, György Csertán, András Balogh, Wolfgang Herzner, Thierry Le Sergent, András Pataricza & Neeraj Suri, *A SW-HW Inte-*

gration Process for the Generation of Platform Specific Models, Microelectronics (ME), 2006.

- Neeraj Suri, Arshad Jhumka, Martin Hiller, András Pataricza and **Shariful Islam**, *A Software Integration Approach for Designing and Assessing Dependable Embedded Systems*, Submitted to the Journal of Systems and Software (In review), 2008.

- **Shariful Islam** & Hannes Omasreiter, *Systematic Use Case Interviews for Specification of Automotive Systems*, In the Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC), 2005.

- Wolfgang Herzner, Martin Schlager, Thierry Le Sergent, Bernhard Huber, **Shariful Islam**, Neeraj Suri & András Balogh, *From Model-Based Design to Deployment of Integrated, Embedded, Real-Time Systems: The DECOS Tool-Chain*, Microelectronics (ME), 2006.

- Wolfgang Herzner, Rupert Schlick, Martin Schlager, Bernhard Leiner, Bernhard Huber, András Balogh, György Csertán, Alain LeGuennec, Thierry LeSergent, Neeraj Suri & **Shariful Islam**, *Model-Based Development of Distributed Embedded Real-Time Systems with the DECOS Tool-Chain*, In Proceedings of Society for Automotive Engineers (SAE) Aerotech, 2007.

1.5 Thesis Organization

We organize the thesis as follows.

Chapter 2 depicts the thesis background, preliminaries and system architecture and models. We also discuss the related work. The system model mainly consists of HW model, system level partitioning, communication systems, SW model, fault model and the constraints model which are elaborated in this chapter.

Chapter 3 presents our dependability driven system level co-design and optimization methodology. We introduce different criteria for designing dependable RT embedded systems. We present the mathematical formulation of the problem following the design space exploration and the constraints handling techniques. The framework for the design of an integrated embedded architecture comprises two key parts – *(a)* the SW-HW mapping and *(b)* the quantification and optimization.

Chapter 4 describes the dependability and RT driven SW-HW mapping following the prime drivers such as FT schemes, reduction of influence as well as the communication bandwidth and schedulability analysis. We present jobs and nodes ordering heuristics and supporting data structure for doing the consolidate mapping of both SC and non-SC applications. The mapping process is illustrated through a real automotive example.

Chapter 5 presents the MVO framework. First we describe the essential issues relevance to the optimization such as general optimization ideas, properties and selection criteria of variables. As a key contribution the quantification and estimation of the design variables from dependability, RT and resource perspectives is then provided. Finally, based on an existing optimization algorithm, we present the MVO approach.

Chapter 6 provides the experimental set up and based on this set up we evaluate the approach in two folds. First the performance and effectiveness of the generated feasible mapping is evaluated. Second we perform the empirical evaluation of the MVO showing the convergency and quantitative gain. Moreover we provide validation and comparative study of the approach.

Chapter 7 develops a prototype of the system level co-design, which includes transformation based design approach, architecture of the prototype, scheduling tool support and deployment process of a system. We present an overview of the model driven development and platform based design relevant to the transformation based approach.

Chapter 8 extends and adapt the approach by adding the measure of reliability and system level power optimization while considering an architecture/platform comprises nodes of different speeds and failure rates. We present the applicability of the SW-HW mapping process on such an architecture.

Chapter 9 concludes the thesis by providing the main ideas and contributions of the developed dependability driven co-design methodology. Finally, we give some future research directions related to the thesis work.

Chapter 2

Background and System Model

The needed background information, the base architecture and relevant models are described for developing the design concept. This chapter is mainly divided into three parts: preliminaries, related works and system architecture and models. We start by describing target application domains and their requirements where we target to apply our developed design methods. Since the design is relevant to the problem of jobs and nodes assignment, we investigate some existing resource allocation problems including metaheuristics techniques for optimization. Relevant research within the area of design and optimization of dependable RT embedded systems has been exploited. A notable focus has been given on the related work of allocation, scheduling and multi objective optimization since they are the key aspects of our design process. Finally, we present the system architecture which comprises of various discrete models. The system architecture and models furnish the fundamental issues of the co-design methodology presented in this thesis. The system model comprises the HW architecture model, communication systems, system level partitioning, the SW model, constraints model and the fault model.

2.1 Preliminaries

For a better understanding of the applicability of our developed methods and techniques, we further elaborate on the target application domains. The target system requirements extracted from those applications are classified into functional, performance and dependability requirements which are then described. A preview of existing *resource allocation problems* and *metaheuristics* is depicted which are the premises of the heuristics based mapping and optimization.

2.1.1 Target Application Domain

As mentioned earlier, we target applications from a wide variety of domains like automotive, avionics and control. We categorize a system domain consisting of SC and non-SC applications. If we can classify any domain under these classes, they can be designed according to our developed method and process. The SC applications are the ones which have to provide services despite of occurrence of faults. This category of applications must be highly dependable and must have to provide services within a certain time period otherwise any malfunctioning or delay produces catastrophic consequences on the system. The non-SC applications can be categorized as the soft RT systems where presence of any perturbations or missing a deadline though degrade the system performance do not lead any catastrophic consequences. The reason for considering both types of applications is that both are present mostly in large heterogeneous FT-RT embedded systems. These systems are classified as hard RT and soft RT systems [43], which usually corresponds to the SC and non-SC systems respectively.

Therefore the target applications of the developed co-design methodology are highly safety- and mission-critical and must be able to satisfy stringent dependable and hard RT constraints. Furthermore, non-SC applications are targeted as well. Though our developed method can be applied to various applications domain, for illustration and experiments we use applications mainly from the automotive arena in order to provide the proof of applicability of the concept on a particular domain.

2.1.2 System Requirements

A typical FT-RT embedded system has to comply to a set of frequently contradicting requirements formulating both envisaged functional and extra-functional (performance and dependability) characteristics. Consequently, a typical set of requirements for dependable embedded system should cover the following:

Functional Requirements

The *functional* requirements simply describe the operational profiles or the functional specification of applications as well as the operation of application jobs. Each function takes a set of inputs and produces a set of outputs after performing a successful computation/operation. Therefore these requirements specify the expected services, functionality and characteristics of the system. These types of requirements entail the phenomena that the out-

put of the function/application should comply to the specification. At job functionality level we consider this requirement as a black box with input and output ports to receive and send the necessary data and services among them.

Performance Requirements

The *performance* requirements describe all timeliness requirements and properties of RT systems. The system particularly the hard RT systems [43; 44] must have to respect all these requirements in order to deliver completely predictable and deterministic services in addition to the compliance to the functional specification. For instance, applications must finish their execution and provide services within a certain temporal limit, i.e., within its deadline. The deadline of the application or job has to be set under this requirement. These characteristics must be shown even in the presence of undesirable circumstances or any external perturbations. The performance requirements impose the condition that the output of a job should be correct in time domain. The output of a job depends on its execution/computation time and the dependency with other jobs which controls jobs starting time. This dependency is known as the precedence relation between jobs. A successor job can only start computation once it gets data from the predecessor job. The computation time of a job depends on its functional complexity and the speed of the processor where it is executed. This does not depend on how the jobs are mapped onto nodes. The computation time may vary at run time due to many reasons. For example, it may vary due to different conditional branches which depends on different inputs. However in order provide a deterministic behavior for the SC hard RT systems, this time should be fixed a priori with their worst case execution time (WCET) – the maximum time a job may take to perform its execution. The system designer either assumes these properties relying on the experience on complexity of system functionalities and the type of processors used to host the jobs or can obtain by WCET analysis [45; 46]. A constraint imposed on timing behavior of a job or an application is called as timing constraint.

Dependability Requirements

The *dependability* requirements consist in general any one or more of the aspects of dependability properties, which are reliability, availability, safety, security, integrity and maintainability [3]. The dependability of an overall system should be much higher than the dependability of its individual SW, HW components and other resources. In case of replication based safety, the

top priority requirements relates the number (or cumulated reliability) of replicas to the designated reliability of the system. For example, if dependability of an individual component is 99% (in terms of reliability measure) then the dependability of a system consisting 4 components is 96.06% and for 50 components it reduces to 60.50%. If a system composed of 250 components even with higher dependability of each component equal to 99.9%, the overall system dependability reduces tremendously to 77.87% provided that there have been no techniques or redundancy applied to improve the system dependability. The dependability requirements imply that the output result of a function of both value and time domain must be correct despite of any internal and external perturbations. Hence, vigorous techniques need to be applied to achieve a desired level (set by the users) of overall system dependability. However every function in a system must not need to achieve such a level. The more complex and critical the function is the higher the level of dependability requirements.

All the functionalities or jobs are not equally important in a system. The importance or the criticality of a job is a positive integer number depends on how critical the job is. For example, in cars the brake force control job or the distronic function which measures the distance of an in-front car is more important than the window control job. In a fly-by-wire system the job that controls the flight of the flight control and management system is more important than the navigation system which determine the current relative position of the aircraft. The cabin air flow and temperature control jobs are more important than the functionalities which run entertainment systems onboard and so on. This way the system designer can set the degree of importance or the criticality degree of jobs or applications. Moreover the design decision has to take into account any erroneous influences from the lower critical jobs to the higher critical ones. IEC 61508 [47] introduces a safety integrity level (SIL) for functions safety, e.g., there is a standard for automotive and control applications. Every safety function is associated

SIL	Criticality	System failure	Probability of dangerous failure per hour
4	Safety critical	Catastrophic failure	$10^{-8} \longleftrightarrow 10^{-7}$
3	Safety relevant	Sever failure	$10^{-7} \longleftrightarrow 10^{-6}$
2	Critical	Major failure	$10^{-6} \longleftrightarrow 10^{-5}$
1	non-Critical	Minor effect	$10^{-5} \longleftrightarrow 10^{-4}$
0	no dependability requirements		

Table 2.1: Safety integrity levels

with this level shown in Table 2.1 according to its criticality which is then taken into consideration in the design process. The degree of criticality of an application or a job depends on its SIL level, the higher the level is the higher the degree of criticality. The application jobs are replicated according to the degree of criticality in order to provide FT.

2.1.3 Preview of Resource Allocation Problem

Since we formulate our allocation problem as a *constraints satisfaction problem* (CSP) [50], we discuss some existing resource allocation problems (as follows) relevance to the jobs and nodes assignment problem. Each assignment problem either generates a feasible solution while it satisfies the defined constraints or generates an infeasible solution when it does not. Moreover the solution is said to be optimized if it minimizes or maximizes one or more particular objectives.

Multiple Knapsack Problem

The multiple knapsack problem [48] is described with a set of items of given sizes (each item has an associated profit and weight) and knapsacks (bins) of given capacities such that each of the knapsacks is filled with a subset of the items without exceeding the knapsack capacity (a feasible assignment) and maximizing the profit (optimization). The multiple knapsack problem is a generalized case of well know single knapsack or bin pack problem and similar to an special case of generalized assignment problem.

Generalized Assignment Problem

This type of assignment problem [48] is similar to the multiple knapsack problem described above where the profit and size of the items can vary based on the knapsack it is assigned onto. The generalized assignment problem is NP hard and often described in terms of assigning tasks to agents, assigning jobs to machines and similar types of assignment problems. The agent has a given capacity and the task has profit and weight associated to each of the agents. The goal of this assignment problem is to disseminate all the tasks among the agents such that the sum of weights of all tasks assigned to a particular agent does not exceed the capacity of that agent and the total profit is maximized. Each task is assigned onto exactly one agent and the assignment problem is constrained with the availability of resources.

Timetabling Problem

The university course timetabling [49] is the problem of assigning time slots and rooms to the classes or lectures satisfying a set of rules. Rules are defined as constraints for instance two lectures should not be allocated to the same room at the same time or two lectures should not be assigned to one professor at the same time.

Job Shop Scheduling Problem

The job shop scheduling problem [50] is represented by a set of jobs and resources (processors) where each job is associated with a set of operations or activities. Each activity needs a certain amount of time to finish its operation on a particular processor. The objective is to find an scheduling policy assigning each job to one resource and specifying the operation of each job starting and finish time onto that resource/processor such that overall time required for all jobs is minimized.

Traveling Salesman Problem

Instead of an allocation and constraint satisfaction problem like those described in above, this problem rather directly deals with an optimization problem. The traveling salesman problem [51] is given with a set of cities and a traveling cost associated between any two cities. The goal of the salesman is to visit all the cities once and get back return to initial starting point by minimizing the total cost.

All these problems discussed above are NP hard and generally difficult to solve in a tractable way. Therefore these kinds of problems usually need heuristics and metaheuristics based approaches and their guidance to find a feasible and optimized solution. Often these problems are formulated as constraints satisfaction and multiple objective or combinatorial optimization problems. These problems are relevant to the job and node assignment problem described in this thesis.

2.1.4 Metaheuristics in Optimization

Metaheuristic or hyperheuristic is an improvement heuristic method for solving a class of constraints satisfaction and combinatorial optimization problems. Thereby, generally metaheuristics are applied to complex problems for which there is no satisfactory problem specific algorithms or heuristics, or practically sound to implement such a method. Exact algorithms which search for optimal solution are computationally very expensive while

metaheuristics algorithms produce *near-optimal* solution within a reasonable time [52]. According to [53], the principle of optimality states that – *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.* The metaheuristics are somehow works obeying the same principle. Metaheuristics guide the search process and efficiently explore the design spaces in order to find a near-optimal solution or a set of solutions. Investigations show that these types of optimization algorithms are suitable to find design solutions for RT embedded applications since usually they are not problem specific. In the next chapter we formulate our design problem mathematically as constraints satisfaction and multi objective optimization problems. Though there are various rigorous metaheuristics techniques exist, in the following we describe a very commonly used and most successful ones presented in the literature [52; 51; 55; 56].

Simulated Annealing (SA)

Simulated annealing is a metaheuristic based algorithm which applies computational stochastic strategy to find a near optimal solution. The goal is to find minimal configuration of the states of a system, e.g., to select the lowest point in an energy landscape by minimizing an objective or energy function. The method was derived from the observations of how slowly the cooled molten metal can result in a regular crystalline structure, i.e., a so called annealing process. It has been first successfully applied to a combinatorial optimization problem by [51]. The algorithm starts with an initial state/solution and progressively explores all the states. The initial solution is replaced by the candidate (a different state) solution if the candidate solution has lower energy value than the initial. It accepts or rejects the candidate solution with higher or equal energy value depending on the value of a probability function, termed as *acceptance probability* function a_p. The value of this function is controlled by a so called temperature parameter and thereby implement a *cooling schedule*. Initially this temperature is set to a high value to accept some solutions with higher energy values so that it explores large search space. Otherwise, if the temperature is set to a lower value the solution may get stuck into a local optima and it may not converge to a global optimized solution. The probability of accepting the solution of higher energy value is decreased with the decreasing of temperature value, i.e., with the progress of search. Eventually the system state reaches to a lower energy value point. The cooling schedule controls such phenomena. During the search it always stores the value with the minimum energy. The algorithm has been proven, e.g., by [54] to *converge* asymptotically to a global minima.

Tabu Search

Tabu search is a metaheuristic approach which employs some specific strategies (*short and long term memory, intensification and diversification*) to guide the search in a systematic way in order to find an optimized solution. The algorithm implements these strategies in order to be able to avoid the local minima and to efficiently explore the design space. The tabu search ideas were first implemented by [55]. The short term memory implements a short list of *tabu* solutions that have already been visited and prohibit the further search to select solution from that list. Long term memory keeps the history of the over all search process in order to be able to collect the information of good visited solution. It is a kind of learning process which generates other two strategies of intensification and diversification. The intensification allows to use the accumulated search experiences and diversification refers to exploration of new search spaces. The latter enables to explore the unvisited space and generates the new solution which differ from the old ones. The former exploits the good visited solution to create the neighbors. The proof of convergency of tabu search algorithms is difficult.

Evolutionary Algorithms

Evolutionary algorithms are the population based search methods inspired by the principal of natural evolution of survival of the fittest. They differ from other metaheuristics in the sense that in each iteration they perform a search from a set of solutions (population) rather than a single solution. A number of operations is applied to the individual solutions of the current population to generate the individuals of the population of the next generation. *Crossover* and *mutation* techniques are used as operators. These operators are also called genetic operator. Crossover refers to produce new individuals by recombining two or three individuals. Mutation refers to the small change of single solution and add diversity to the population. The selection of individuals are based on their fitness (usually implemented by objective functions). The higher the fitness the higher the probability of individual that it gets into next generation/iteration population.

Genetic algorithm [56] is a type of evolutionary algorithm which focuses on optimizing general combinatorial optimization problems. It follows the same process as in the evolutionary algorithms and has the capability to evolve the population and then converge to a higher quality solution. Though in general the evolutionary algorithms are often applied for global optimization, the convergence to a global minima/maxima is only guaranteed in a weak probabilistic sense.

2.2 Related Research

This section describes existing research in the area of designing dependable RT embedded systems which are relevant to this thesis work. We particularly focus on system design and co-design methods, the mapping problem (allocation and scheduling) and on the design optimization (single and multiple objectives). These are key problems which need to be solved for the design of FT-RT embedded systems.

2.2.1 System Design and Co-Design

First, we discuss some existing works on SW-HW co-design which focus on system level design. An overview of recent past co-design work can be found in [57]. The author has mentioned that co-design is an ideal way to explore the design space and to make efficient trade-offs between SW and HW. Previous co-design approaches have relied exclusively on the mapping step to achieve optimality where the HW and SW design tasks are optimized separately (proper co-design is not performed) using different techniques and then integrated to the target architecture [8]. The co-design community in general, referred to SW-HW trade-off evaluation and implementation at a very late stage of the system development process. The current techniques are not sufficient to meet future system design demands. Moreover, many of existing techniques and tools focus at lower level design (in the absence of abstraction) and use a single processor architecture for smaller embedded systems such as COSYnthesis for eMbedded micro Architecture (COSYMA) [9] – chapter 8, LYngby CO-Synthesis (LYCOS) [58]. COSYMA is later upgraded with the optimization of multiple objectives based on simulated annealing [59]. A unified co-design tool POLIS [60] supports the use of a multiprocessors architecture and guides the user from the design of specification to the implementation. Mostly, the existing work still follow the *fix-it-later* philosophy, i.e., faults or errors occurred in the design are detected and fixed at the late design stage. A very limited set of trade-offs, optimization and design refinements can be done at the low level since most of the crucial function and platform decisions would have already been made [7] – page 2-3. If the design processes are carried out at such a level then when any modification is necessary, e.g., fixing any faults, will result significant cost and time.

In the recent past, many researchers have focused on the design at a higher level of abstraction, i.e., at system level such as [8; 26; 61; 62; 63; 64; 65; 66]. The function-architecture co-design and optimization methodology of embedded systems at higher level of abstraction is addressed in [8]. The author elaborates on the function/architecture trade-offs and optimization analysis

of heterogeneous control flow dominated systems for both data and control flow applications at system level. A mapping and optimization analysis of heterogeneous embedded systems for distributed applications is discussed in [26]. A system level synthesis comprises of selection of the architecture including general purpose and dedicated processors is presented in [61]. The process is termed as the allocation and mapping of specification onto the selected architecture in space (binding) and in time (scheduling). Author uses evolutionary algorithms for selecting the architecture with a heuristic based scheduler for the synthesis of data-flow dominant SW and HW systems. [62] develops a Y-chart based system level simulation model for design space exploration and multi objective optimization for embedded multimedia system architecture. [63] explains various aspects of system level design and stated that the co-design of HW and SW in diverse systems may lead to productivity gains, lower costs and first-pass design success. In [64], reliability is addressed in a system level co-design flow. Author first considers the satisfaction of timing, cost and area constraints and then introduces reliability in a second level of the design which may produce inefficient design with respect to the first level criteria. Co-design techniques which particulary implement automotive applications on a federated architecture is presented in [65; 66]. [65] mentions about increasing the number of ECUs with increasing applications and gives the future directions of using an integrated architecture for automotive systems.

The existing design techniques for embedded systems often either do not systematically consider the metrics such as dependability, safety and RT or consider them individually at different design stages making the system design not dependable enough and costly. Particularly, the needed focus has not been given on dependability and its optimization while designing such systems. Instead, we put the main focus on the aspects of dependability. To the best of our knowledge, this is the first work which presents a dependability driven system level co-design methodology of embedded systems while considering other properties such as RT, resource utilization and power consumption. Analysis and trading-off between SW and HW at system level reduce the number of iterations at different design stages as well as reduce the time-to-market.

2.2.2 Mapping – Allocation and Scheduling

In the first phase of the design process we create a feasible mapping by allocating HW resources to jobs while satisfying several constraints including separation of replicas and timing. Different techniques have already been used for solving such resource allocation problem, e.g., constraint program-

ming [67; 68], branch and bound [69; 70; 71], inform branch-and-bound and forward checking [37; 68] and mixed integer programming [72]. All these approaches perform the mapping (allocation and scheduling) straightforwardly applying the above mentioned techniques. [69] describes the replication and the allocation of task modules (including replicas) onto processing elements (nodes) satisfying the precedence and deadline constraints for distributed RT systems. Replicated tasks are disseminated over different processing elements and the probability of satisfaction of deadline is termed as the probability of no dynamic failure. The author uses the term probability of no dynamic failure as an objective. An static allocation of periodic task modules onto heterogeneous processing elements (processors have different speeds) is presented in [70] for distributed RT systems. Mostly tasks are selected to assign on the processing element of higher speed. This approach is generalized in [71] for both tasks and messages assignment and scheduling. [69; 70; 71] employ a branch and bound algorithm for the allocation problem in order to efficiently explore the search spaces.

A disadvantage of these approaches is that usually they do not put additional efforts to reduce the search space a priori while solving the complexity of the problem. Many of them are limited to handle only a few constraints either from RT or from FT. [67] addresses the allocation and scheduling problem satisfying multiple constraints. The author applies symmetries exclusion to reduce the search space as an add-on technique in their approach. However this is more desirable in a homogeneous architecture where the architecture comprises nodes of symmetric multiprocessing (SMP). It is required that the nodes should be unused to employ them in the symmetry exclusion technique. An enhancement of the Quality-of-Service (QoS) based resource allocation model [72] is presented in [38], where a hierarchical decomposed scheme by dealing with smaller number of resources is described enabling QoS optimization techniques for large problems. Tasks replication is used as a QoS dimension to provide FT. In this thesis, we decompose our approach into several subproblems and phases in order to reduce the complexity of solving the problem. We also provide techniques for satisfying the design constraints such that search spaces can be reduced. We describe more on this in Section 3.4.

The major requirements for designing embedded RT systems are to meet RT requirements and to provide dependability (achieving FT and avoiding error propagation). Commonly used approaches typically address RT and FT on a discrete basis [37; 38]). AIRES (Automatic Integration of Reusable Embedded Systems) [37; 73] describes the allocation of SW components onto HW platform for RT and embedded applications satisfying multiple resource constraints. Based on constraint programming (CP), [68] presents an ap-

proach for solving the constraint-driven resource assignment and scheduling problems for system level design. They develop a Java based constraint solver engine which satisfies a set of different types of constraints including power and the framework is extended by additional constraints. However dependability/FT is not considered in any of these approaches. Moreover, when scheduling for RT systems is performed, a predetermined allocation or a simple allocation scheme is used (e.g., [20]). If the scheduling is performed without assuming any pre-allocation it may significantly increase the computation complexity and can make the problem intractable (cannot be solved in polynomial time [35]). Also if the allocation and scheduling are considered completely separately, important information (e.g., considering constraints) used from one of these activities is missed while performing the other. On the other hand, usually FT is applied to an existing scheduling principle such as rate-monotonic [74] or static off-line either by using task replication [75] or task re-execution [76; 77]. [75] addresses timeliness and dependability properties at the task level of scheduling. Utilizing the existing slack from the scheduling, [76] implements FT mechanism where tasks perform the re-execution in presence of transient and intermittent faults. Similarly, a static off-line contingency scheduling approach with recovery technique for tolerating transient faults of SC embedded systems has been provided in [77].

Existing all these approaches typically do not address all the constraints neither use any specific constraints satisfaction techniques or use a limited fault model where dependability is essential. We apply *constraints prioritization* [78] during the allocation phase in order to satisfy the constraints which reduces the complexity (takes less iteration to find the solution) to solve the problem. [79] specifically addresses dependability driven mapping (focuses on minimizing interaction) and presents heuristics for doing the mapping. However the focus is on design stage SW objects to aid integration. A survey of various SW development processes addressing dependability as extra-functional requirement at both early and late phases of the design is described in [80]. The paper discusses several projects where most of them consider both functional and extra-functional requirements particularly FT in their design process. The author mentions that taking extra-functionality at the late stage of design make the design more complex and costly. For example most of them used an ad-hoc based solution in order to provide dependability where FT is added once the main functionalities of the system have been implemented often results more resources, complex system structure, poor performance and high cost. Unlike other approaches for task/job allocation, the focus at this stage of our work is on finding an initial schedulable allocation, a so called off-line allocation of mixed critical applications onto a distributed computing architecture. In our design process we consider

both functional and extra-functional properties at the *system level*, i.e., they are addressed and analyzed at the early phase of the system level co-design.

2.2.3 Optimization

Many research efforts have already been made on design optimization of embedded systems. The focus did rely on RT embedded systems domain where satisfaction of FT and/or RT constraints and optimization of a single objective has widely been studied [75; 76; 81; 82; 83; 84; 85; 86; 87; 88]. Satisfaction of timing constraints and minimization of total completion and communication times are addressed in [81]. [82] considers the maximization of the probability of meeting job deadlines as an objective function while allocation (assignment and scheduling) of periodic tasks onto distributed platform consists of different processing nodes. The paper discusses about the precedence and deadline constraints of tasks. An static optimal scheduling for both tasks and messages can be found in [71]. The author minimizes the maximum task lateness (difference between the task completion time and the task deadline). Generation of a feasible non-preemptive scheduling for distributed static systems is presented in [83], where the author minimizes the jitter for periodic tasks. The author explains the application of simulated annealing for distributed RT scheduling and optimization instead of using conventional search algorithms. Satisfaction of multiple constraints (timing, FT and memory) and optimization of a single variable (bandwidth utilization) is presented in [84]. The paper also applies simulated annealing for finding both the feasible and optimized solution. An additive objective function is used including function for satisfying hard constraints and minimizing the bandwidth utilization. However we may often face difficulty to poise these two factors (constraints satisfaction and meeting objectives) together specially when setting the weight/trade-off factors in the objective function. All these approaches either consider FT as constraint or optimize a selected operational variable from an RT perspective. Task allocation for maximizing reliability hence enhancing dependability has been presented in [86; 87], where failures from processor and communication links are considered to measure the system reliability without redundancy. Author assumes processors and the communication links with different failure rates. [88] presents a hybrid particle swarm optimization based algorithm for solving the task allocation problem with the goal of maximizing the system reliability subject to resource constraints. The overall objective function includes both the constraints satisfaction by adding a penalty function and a single objective function for optimizing reliability. [93] elaborates a reliability model for tolerating one processor failure while satisfying RT constraints for peri-

odic tasks. In [85], author describes that the interplay of active replication and re-execution can provide an optimized design from the scheduling length point of view. The presented approach assigns FT policy to the processes which are mapped to the processors such that transient faults are tolerated and timing constraints are maintained. A tabu search based algorithm has been used for optimizing the mapping of distributed FT-RT systems.

Though optimizing one variable is straightforward, optimization of multiple variables while satisfying multiple constraints is a more complex and difficult problem to solve. Simultaneous optimization of several variables entails hard trade-offs between each other and is considerably more difficult to find a solution. Several mapping techniques do exist but few are concerned with optimizing extra-functionalities such as dependability/FT and RT issues together. Bi-criteria objectives have recently been attracted attention by many researchers such as [91; 92; 93; 94; 95; 96; 97; 98]. Mainly, the optimization involved two criteria, either on FT/reliability and schedulability [91; 92] or on FT/reliability and power [94; 95; 99] or on RT/schedulability and power [96; 97; 98] while providing FT. [91] presents an algorithm based on greedy list scheduling heuristic which maximizes the system reliability while minimizing the scheduling length. An active replication based FT scheme is used to improve the reliability and the scheme is represented by the reliability block diagram that ease the analysis of reliability. A target/requirement value is set for both objectives and the algorithm iteratively improves the solution towards the target values. The author uses a heterogeneous architecture where processors are connected by point-to-point communication links. A genetic algorithm based allocation of system resources to the applications with the goal of minimizing the execution time and the failure probability of applications is discussed in [92]. Like the previous paper a same type of heterogeneous architecture is used. The idea for maximizing reliability was on assigning the highly critical tasks/jobs on less failure prone processors, jobs require long computation time onto a fast processor and providing more replicas to increase the reliability.

We now discuss some related work on optimizing power while considering reliability and schedulability. [94] presents an approach to utilize the scheduling slack for power saving while preserving the overall system reliability. The author deals with a problem of allocating slack to minimize the power consumption and to maintain the reliability. They use a dynamic voltage and frequency scaling (DVFS) method to reduce the processor frequency and consequently reduce the power consumption. An energy efficient optimistic TMR (triple modular redundancy) scheme is discussed in [99] where one processor is slowed down or put in sleep state to save power provided that if there are any faults or errors in any other two processors then the third

one should be able to speed up the execution and can finish the execution by the deadline of the application.

However concurrent optimization of more than two criteria is still evolving in the area of dependable embedded systems. As we mentioned very few addresses more than two objectives at the same time. Relying on pareto based optimal solution and weighted sum methods, [33; 89; 90; 100] focus on optimization of multiple objectives. Multiple objectives are combined into a single weighted function. The focus mainly was on how to optimize several variables instead of the necessary detailed quantification of each of the considered variables. A multi-criteria schedulability analysis based on earliest deadline first is presented in [100]. The author presents an approximation algorithm and showed that the multi criteria problem can be solved in polynomial time. [89] considers several variables for the optimization such as memory utilization, CPU utilization, communication and availability of system services. The variables are then combined as a weighted sum method and solved by using the multi objective optimization algorithm. A multi objective optimization using genetic algorithm is described in [33], where dependability is considered as an optimization criterion together with time and cost for solving the problem. Replication is introduced at tasks level according to their importance to provide the system dependability. However they allow tasks of different applications to be assigned onto processors without providing fault containment. Moreover a generic system design framework for mapping and optimization is not provided. The approaches do not provide detail quantification neither enhance dependability nor consider power as an optimization criteria. The author of [101] discusses multiple objectives such as minimizing communication, load balancing, minimizing the maximum lateness and minimizing the energy consumption. *Load balancing* is defined as the proper distribution of tasks among the processors such that each of the processor is loaded with almost equal amount of computation. The author compares application of different existing algorithms into their approach for example simulated annealing and constraint programming. An approach for the design of RT embedded avionic system is presented in [90]. The author uses a value function within simulated annealing framework and considers several non-functionalities while optimizes different multiple objectives. For an automotive embedded system, [102] provides a multi objective optimization approach for the allocation of functions onto ECUs. The author focuses on optimizing communication load on the bus, weight, cost and power and uses an ant colony optimization metaheuristic technique for solving the problem. The author does not consider any dependability/FT issues.

Nevertheless, optimization of dependability has not yet been focused enough with other multiple optimization variables. Overall, in design op-

timization of FT-RT embedded systems, there is a dearth of work that address and optimize dependability together with criteria such as RT, resource utilization, power. In this thesis we consider optimization of several competitive variables/objectives for designing dependable embedded systems. One of the key focus has been given on the quantification of variables specially estimating *influence* from a dependability perspective. Given the upper most priority on dependability and for simplification, we first consider optimization (minimization) of three different variables: influence, scheduling length and bandwidth utilization. We progressively extend the approach and add the quantification of reliability and power consumption.

2.3 System Architecture and Models

The dependable embedded systems design and optimization framework developed in the next chapter is based on the system architecture and models presented in this section. The system architecture and models mainly comprises the distributed HW architecture model, the SW model, communication model, constraints and the fault model. The *architecture model* describes the HW platform which contains several computing nodes and their resources. The *SW model* describes the functional and extra-functional requirements of application jobs and the HW platform is the physical execution environment for those jobs. The *communication model* establishes the communication system between different nodes of the platform. The communication system is necessitated by the distributed nature of applications and communicating jobs. The *fault model* depicts the types of faults and their causes whereas constraints restrict the possible solutions. A set of functional and extra-functional constraints have been defined which are later then satisfied during the allocation of jobs onto nodes. As a part of the architectural model different aspects of system partitioning, policies for creating robust partitioning and separation of concern between SC and non-SC applications are described. SW reusability is then briefly presented. The classification of influences at application and at job level is also addressed in this section. Through the rest of this section various important aspects and characteristics of different system models are elaborated. All these models are comprehensively taken into consideration in the design issues presented in the next chapters.

2.3.1 The Architecture Model

The system level co-design usually cannot achieve optimality by making design optimizations and exploring suitable implementation choices without ad-

equate specification and proper modeling of the target architecture. Taking this in consideration, our architecture model constitutes a distributed shared HW platform with a network topology allowing every HW node to communicate with each other node. The Figure 2.1 shows the high-level model of the target architecture and resources elaborating the partitioning concepts and the application execution environment where nodes are connected through a bus topology. A HW node is a self-contained computational element and may

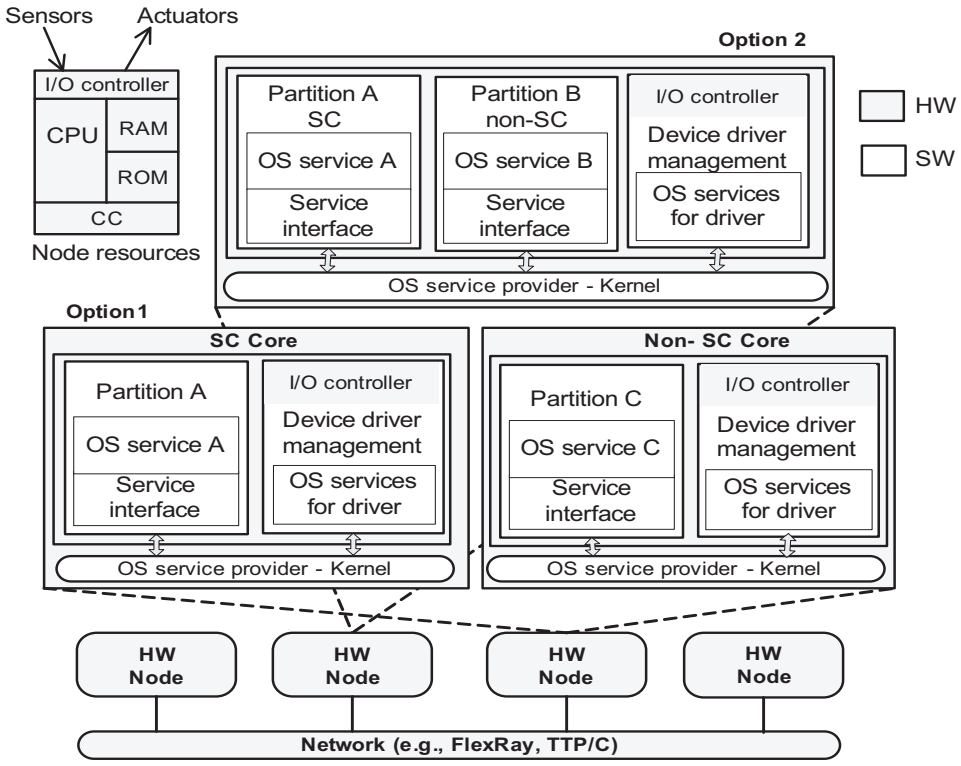

Figure 2.1: High-level model of the target HW architecture

contain a single or multiple processors or a processor with multiple cores (e.g., dual core processors as in *Option 1* in Figure 2.1). The nodes are connected to the network through *communication controllers*. The communication controller (CC) controls the exchange of messages with other nodes. HW nodes may contain additional resources, e.g., *sensors and actuators* which access the CPU through I(input)/O(output) interfaces. The platform may also contain dedicated HW such as digital signal processor (DSP), field programmable gate array (FPGA), application specific integrated circuit (ASIC) to speed up the execution of some particular functionality if necessary. It is not necessarily the case that every nodes of the platform need to have these additional

resources (sensors, actuators, dedicated HW) entailing a heterogeneous nature of the HW architecture.

The set of nodes $\mathcal{N} = \{n_1, ..., n_k\}$ can be modeled as an interconnection *HW graph* that represents limited HW quantity provided by the node processor. The measure of limitation can be in time (e.g., a certain amount of CPU time is assigned) or in space (e.g., a certain memory region is assigned to a partition). The selection of platform or the HW instances depends on parameters such as computation, type of CPU, power consumption, failure rate, size and cost.

In order to achieve strong partitioning between SC and non-SC applications, a node architecture can have the physical partitions, each containing a processor or a processor core, one hosting SC applications and the other hosting non-SC applications (*Option 1*). Another architectural possibility is to have a shared processor running mixed criticality applications (*Option 2*), as long as strong partitioning is ensured (typically implemented by the OS or by other mechanisms). Both implementation choices for platform nodes are illustrated in Figure 2.1 and marked as *Option 1* and *Option 2*. Our approach requires that nodes utilize a similar configuration, either *Option 1* or *Option 2*. In Figure 2.1, the OS service provider (kernel) layer is used to virtualize the CPU, dividing it into protected partitions (shown as A, B, etc.), inside which a job executes. The service interface encapsulates the OS services to the specific job running in that partition. The OS kernel layer supports the *intra-node processor communication* (e.g., by using shared memory or buffer). The communication between jobs residing on the same node is called as intra-node or intra-processor communication. When jobs are residing on different nodes and needs to communicate then the *inter-node communication* (e.g., by message passing) is established. In this case nodes share the same communication channel to send and receive messages. In the next section, we describe more on the communication system.

On Uses of Multi-Core Architecture

In this thesis during the illustration of the mapping process (Section 4.5), we employ a HW architecture where nodes contain dual core processors that have some additional benefits apart from the *fault containment*. We use different cores for implementing SC and non-SC applications which provides the separation between these two types of applications. A particular embedded applications would also benefit from a faster and more repeatable RT response. It is possible to run such RT jobs on a dedicated execution core without interference from other jobs that would otherwise compete for CPU resources. The approach can significantly improve the determinism of the RT

response by reducing its interaction with less time-critical system functions. In automotive applications, an important benefit of multi core architecture is that it provides redundancy to the critical applications. Moreover platforms that use dual-core processors can be constructed with a single processor board containing a single set of central processor resources (memories, disks, and so on). For the OEMs, this could reduce system cost and size by up to 40 percent or more when compared to high-performance RT embedded systems implemented with multiple linked computer systems [103]. The dual-core system architecture is simpler and more reliable compared to segregated systems having their own power supplies, mass storage devices, system packaging, and high-speed communication interconnect hardware. Multi-core platform also benefits from *power consumption* view (since each core runs at a lower frequency) than a single-core processor for the same level of performance. Nevertheless, it may suffer from a single point of failure, e.g., occurring a fault on common and shared resources can damage the operability of both cores. This can be avoided by using less error prone shared resources and components in the architecture.

2.3.2 The Communication Model

As mentioned in the previous section we assume a network topology allowing every HW node to communicate with all other nodes $(n_1, ..., n_k)$. The HW nodes share a network, i.e., use the same communication channel for inter-node communication to send and receive messages (e.g., by *message passing*). Use of different functionalities such as critical, control and multimedia functions in vehicles and competition among automotive companies for using low cost networks have led to design and integrate a large number of competitive communication networks. The communication medium is a critical resource of distributed embedded systems, since loss of communication messages results in the loss of overall system services. Therefore the dependability of the communication system should be an order of magnitude higher than the dependability of the individual nodes [43] – page 33. As our prime focus is on designing dependable hard RT systems, the communication between nodes is primarily based on a *fault tolerant time-triggered network*. A survey and comparison of such networks for SC systems can be found in [104]. Currently for the integration of XBW applications and other control applications (e.g., engine control, cruise control) only these types of FT time-triggered networks are being used. Moreover, to increase the dependability of messages communication within an application, it is possible to use virtual network services [105]. A *virtual network* is an overlay network implemented on top of the physical networks. This network encapsulates the

communication activities between different applications.

The physical network uses the time division multiple access (TDMA) protocol (as shown in Figure 2.2) as its media access scheme which provides deterministic access to the medium. The order of the periodic message transmissions are defined statically at design time. Each node can send messages

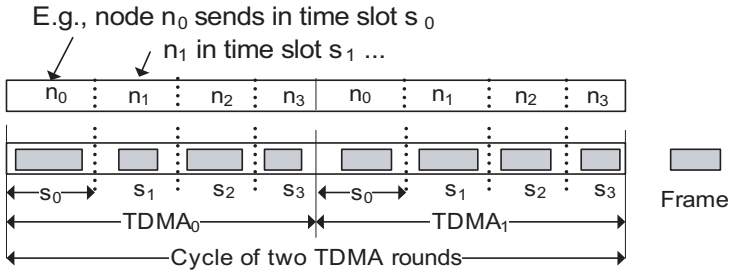

Figure 2.2: A TDMA communication model for TTP/C

only during a predetermined time interval, called slot s_i, over a TDMA round $TDMA_i$. Each node is statically allowed to use a single and fixed slot during a TDMA round, e.g., in TTP/C [16]. A cluster or a communication cycle which is executed periodically is forming a sequence of multiple TDMA rounds. A more flexible message transmission is provided by FlexRay [17]. In the FlexRay communication system a node can use more than one slot to transmit messages in a TDMA round and leave dynamic window with mini slots for the transmission of event messages as shown in Figure 2.3. A

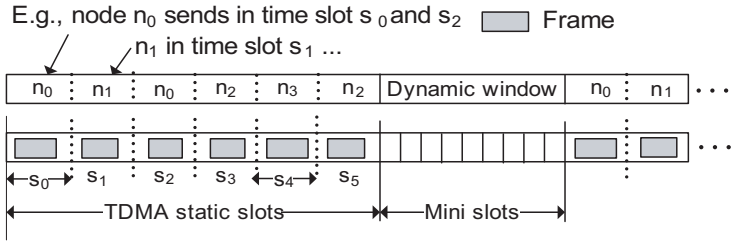

Figure 2.3: A TDMA communication model for FlexRay

network integrating a set of services (e.g., predictable message transmission, clock synchronization, node membership and redundancy management) is necessary for the implementation of dependable RT systems. TTP/C is a purely time-triggered network whereas FlexRay and TT Ethernet support a combination of both *time-triggered* (static window) and *event-triggered* (flexible/dynamic window) communication systems. The time-triggered messages are the ones which appear periodically in the system, e.g., current speed

value of a car. Whereas the event-triggered messages appear in the system in response to an event occurring, e.g., the difference between speeds when it goes higher than a certain threshold.

2.3.3 System Level Partitioning

In this section we describe the aspects of partitioning. We start by defining partitioning which conceptually means that the boundaries among jobs as well as among applications are well defined and protected so that operations of a job will neither be disrupted nor corrupted by the erroneous behavior of another job [32]. This erroneous behavior of a job can be the result of a SW fault or a failure in a HW element used exclusively by that job. Partitioning is needed to ensure that SC applications are not affected by the erroneous behavior of non-SC applications. Each job is allocated to a single partition as shown in Figure 2.1, providing a certain amount of computation and memory resources and means to access devices. The total number of partitions available on a node processor (and consequently the maximum number of jobs that can be allocated to that node), depends on the availability of resources on that processor. The goal of partitioning is to create fault containment units or partitions such that behavior in one partition is unaffected by the behavior of another partition, including faulty behaviors. The concept of partitioning among SW-Cs in an integrated architecture helps emulate a corresponding federated architecture.

In this thesis we present a weaker notion of partitioning through influence reduction for designing integrated systems. Influence is based on the observation that partitioning and fault containment is a strong requirement. The main means of achieving robust partitioning is the implementation of well-defined and protected damage confinement regions between components assuring a guaranteed blocking of inter-component error propagation. The different policies can be distinguished according to the granularity of the architecture, i.e., the notion of components they apply. This also provides the assurance of fault/error containment at different levels. We discuss several system level partitioning aspects as follows:

Higher-/Node Level Partitioning

This level of partitioning provides physical separation between nodes as in federated architecture. It is a traditional policy adopting the granularity of HW nodes as elementary construction and fault isolation components. Each (usually highly dedicated) HW node runs a single functional component of the system. Similarly, replication is introduced at the HW node level. The

same separation principle is used for isolating the implementations of replicas and SC and non-SC functionalities, as well, by strictly deploying each function onto a separate HW node. Damage confinement isolates faulty components. This paradigm necessitates a high HW overhead due to the redundancy induced by the architectural granularization for fault isolation. It typically results in an architecture composed of at least one separate computing node per each individual function interconnected by a fabric of point-to-point communication links.

Core-/Processor Level Partitioning

Processor or core level partitioning is a modified version of node-level partitioning for nodes embedding multiprocessors for a higher computational power. The loose coupling between processors provides a basic inter-processor isolation to take processors as partitioning units, while sharing the remaining (less critical) shared resources such as external communication channels or other I/O interfaces. This is shown in *Option* 1 in Figure 2.1 – Section 2.3.1. This configuration gives the provision of assigning SC and non-SC applications onto different processor cores on the same HW node so that the influence from non-SC to SC applications are prohibited by design.

However the feasibility of the principle of system composition of dedicated parts is limited in scope for ever increasing function complexity of embedded systems. A large number of HW components interconnected by a complex fabric becomes prohibitively expensive from the point of view of resource usage, power consumption and space/weight. Moreover the high level of redundancy rapidly results in reduced overall system reliability despite the increase in component reliability as induced by technology development. This paradigm necessitates the use of a high-level HW redundancy.

Lower-/Job Level Partitioning

Independent of the approach presented above, there is a need for partitioning mechanisms [32; 106; 107] in each processor/core which restrict spatial interactions/influences (e.g., *error propagation* via shared resources between jobs) and temporal interactions (e.g., starvation of a job caused by another one stealing its processor time, or causing a delay of execution of the later) across jobs. Job level partitioning policies use a more fine granular approach by taking jobs as allocation and replication units. It allows HW resource sharing by the deployment of multiple functions (implemented as isolated SW jobs) onto the same HW component (*Option 2* in Figure 2.1). Safety of the systems is still based on replication of the critical parts.

Isolation of the jobs is carried out by means of the standard support mechanism built-in into most modern processors, such as memory segmentation in the Memory Management Unit (MMU) further enforced by specialized HW and OS [32]. Strict spatial and temporal isolation is provided in platforms intended for SC applications (e.g., TTP) by means of extra HW units assuring FT for each main isolation-related functionality both in the multitasking run-time environment and in the communication infrastructure over shared buses. However in this thesis we provide the notion of reducing error propagation/influence during integration so that the partitioning will be less reliant on the use of OS and other partitioning mechanisms.

SW Reusability

Reuse of existing components is a key approach aimed at reducing development and manufacturing times and costs. An efficient workflow covering all phases of the development process (design, component integration, validation and testing, certification) is one of the key factors in the reduction of development and manufacturing costs and time. Two kinds of possible SW reusability is described as follows.

- As job level partitioning looses the dependence on the level of dedication of the HW platforms, they may increasingly become a generic type, thus supporting the reuse of COTS and other legacy components. Automotive, avionics, control and seaborne systems are representative examples of SC systems relying on a rapidly growing number of *SW-Cs* and a *HW-C integration* system design paradigm.

- Another evolving form of reuse is that of the *intellectual property*. While the change in the functionality offered by subsequently developed members of a product family follows typically an evolutionary path, their implementations can drastically differ due to the revolutionary changes in the HW platform technology background.

Note, that there is an interesting interplay between reusability and robust partitioning in building SC systems [21]. As the core concept of robust partitioning is a strict componentization assigning a single partition to each individual functionality to be executed, modification of a functional component influences only those ones, which are in an explicit functional interdependence with it. As side effect freedom is guaranteed by principle with respect to other ones, robust partitioning facilitates the reuse, modification, debugging, integration, and certification of components.

2.3.4 The Application and SW Model

We consider a SW model consisting of both SC and non-SC applications derived from the automotive, avionic and control domains. Compatible to a system level design method, the SW model is constituted at an abstraction level completely independent from architectural details. Consideration of varied system functionalities from several application domains significantly increases the design complexity. The complexity of large heterogeneous distributed embedded systems can only be managed if the overall system can be decomposed into (nearly independent) applications with linking interfaces that are precisely specified in both the value and time domain [108]. A nearly independent application of a large distributed RT system typically performs a specified service. For instance a brake-by-wire system allows the vehicle driver to electronically control the wheel brake. We use the SW model which consists of such applications ($\mathcal{A} = \{A_1, ..., A_p\}$) of varied functionality and criticality. Applications are subsequently decomposed into smaller units, called jobs. We define a *job* as the smallest executable SW fragment with basic communication capabilities for exchanging information with other jobs. Jobs are the leaf functions of a tree-base functional decomposition [7]. No knowledge of the internal structure of a job is assumed, allowing for the integration of jobs for which no source code is available. This allows us to take the benefit of system level design where a job is characterized by their interfaces hiding their implementation details. Therefore the input and output behavior of jobs and the functional dependencies between jobs are sufficient to know.

We consider a job to have the following properties, which are required as input to the co-design process during the mapping phase:

- **Job name and ID**: Each job is provided with a unique name and ID.
- **Input and output ports**: A job is represented as a black box with its input and output ports. A job employs input ports for using the services from other jobs particularly from the predecessor jobs. The output ports enable a job to provide services to the successor jobs or to provide services to the environment.
- **Timing requirements**: An application as well as its jobs has to finish their functional operations by a certain time. Moreover computation of one job depends on the result of another job which is often described by the precedence relations. This relation also impacts the starting time of a successor job. These properties are defined based on the performance requirements mentioned in the earlier section.

- **Bandwidth requirements**: This property indicates the volume of data between communicating jobs in terms of bytes. The network bandwidth is required only by the inter job communication.
- **Dependability requirements**: According to this requirement, the degree of replication dc_i necessary for the i^{th} job is set in order to provide the required level of FT. dc_i depends on the level of criticality of an application or a job. This property is defined for example applying the technique described in Section 2.1.2. The influence value between jobs is also set under this requirement.
- **Resource requirements**: These requirements are summarized in a record (represented by the vector $\vec{r_i}$) composed of the different quantitative descriptors of resource capacity required, such as memory size, availability of a certain kind of sensors/actuators.

SW Graph

The application model described above can be represented by a SW graph. The *SW graph* consists of a set of jobs and their interactions and communications. Inter-job communication is characterized by a *weighted directed graph* (WDG), $G = (\mathcal{J}, E)$, having the job types as vertices V, and an edge between jobs j_s and j_t, if they communicate. At job level the timing properties is represented as (t_i), which is the triple of $t_i(EST_i, CT_i, D_i)$, where EST, CT, D are the earliest start time, computation time and deadline of a job respectively. EST is the earliest possible time that a job can start its execution. CT is defined as the amount of time required by a job to complete the execution on a particular processor. CT depends on functional complexity of the job and on the speed of the processor it is running on. D is denoted as the deadline of a job by which a job or an application must finish its execution. Based on their experiences the system designer estimates the values of these properties. However the values can be determined for example deadline assignment algorithm as in [109] can be applied in order to set the deadline for jobs as well as for the application. Precedence relations between jobs are utilized to calculate the deadline. $e_{ij} \in E$ is an edge between two job vertices $(v_i, v_j) \in V$, which is the notion of both of influence (I_{ij}) and communication data $(b_{i,j})$ (bytes) between jobs. I_{ij} denotes the cumulated conditional probability of error propagation from the source job j_s to the target job j_t, either via message passing or shared resources, assumed, that j_s is in a erroneous state. An estimation of determining this value is detailed later in the thesis where this property is taken as a design criteria. $b_{i,j}$ is the intensity of the required communication between jobs, for

instance measured by the maximal total size of information to be transferred per execution cycle.

In case a job needs to be replicated for providing dependability/FT, a new node as well as a new communication link is created in the SW graph (as shown in Figure 2.6). The replication depends on how important and critical the job is, e.g., if a job has the highest degree of criticality then it needs to be replicated three or four times and if the job has next degree of criticality then it needs to be replicated two times. This can be set, e.g., according to the SIL level as described above. Replicas themselves are connected by edges of weight 0 expressing complete isolation, i.e., there is no influence among themselves.

Influences Across Modules

Influence is defined by the probability of error propagation from a source module to a target module. Faults can occur either in the source module or in the communication channel. Shared memory is a typical element potentially causing error propagation between different functionalities. This propagation depends on the size of the memory they share and how often they access it. Errors can also propagate through message passing which depends on the size of the sending/receving messages and how frequently messages are being sent and received. All these error propagations between any modules are termed as influence. We distinguish two classes of influences (error propagation), depending on the level of detail at which the system is observed. The classification of influences is shown in Figure 2.4 and is discussed below.

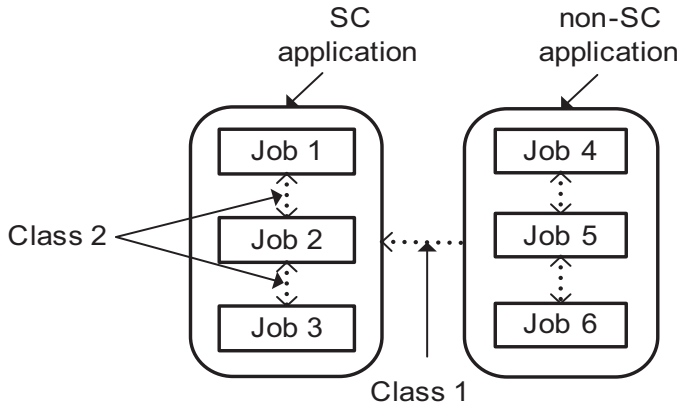

Figure 2.4: Influence across modules at different levels

Class 1: This class of influence is considered at application level. Faults/errors must be detected before they propagate into another SC application. In this case, high criticality applications need to be shielded from low criticality applications, but not vice-versa. For example, it may be acceptable, in terms of safety, for a brake force control application to corrupt the air conditioning systems in vehicles and for a flight control application to corrupt the onboard passenger announce subsystem in avionics systems but there will be catastrophic consequences if it happens in opposite way.

The issue of error containment plays a crucial role since consequences of a particular error must not have an adverse effect on a SC function. If this cannot be ensured, the error needs to be classified as catastrophic [43].

Class 2: This class of influence is occurred at job level. Interactions between two communicating jobs of an application, may lead to propagation of errors from one job to another. When these jobs are placed on different nodes then this influence may lead to an error propagation at node level. Also, the sharing of resources by different jobs on the same node may lead to error propagation.

The interaction/influence at different levels can lead to harmful error propagation from one module to another (influences between applications or between jobs). When the errors are not propagated they are said to be contained within a module (error containment) and avoid catastrophic consequences by avoiding multiple failures. Thus, to ensure fault/error containment, we strive to reduce their propagation probabilities while performing the mapping and its optimization (for details see Section 4.1.2 and Section 5.2.1).

An Example Application Model

We briefly describe an automotive SC application that is a *brake-force control (BFC)* of adaptive cruise control system. BFC is a video-based emergency brake for the collision warning and avoidance system. The BFC is a distributed periodic application. This application is decomposed into four jobs (speed sensor j_1, distance control j_2, brake force control j_3 and brake actuator job j_4) as shown in Figure 2.5. Speed sensor job is responsible for reading car speed value from the speed sensor and sends the speed message data to brake force control job. Brake force control is a control object job for computing the necessary brake force. Distance control job reads distance value of the nearest object from the image sensor and sends the corresponding value via a message to brake force control job. This job now computes

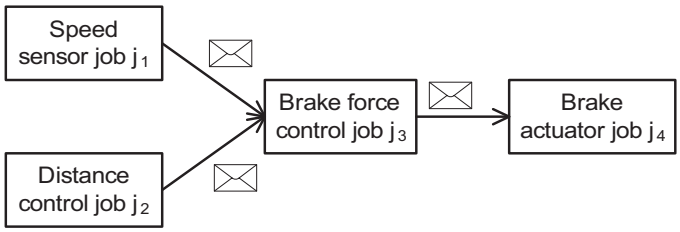

Figure 2.5: Brake force control application

the brake force within a certain period of time and transmits the computed message to brake actuator job which activates the brakes in order to make the necessary actions to avoid collision. In order to provide FT each job needs to be replicated according to their degree of criticality dc_i or according to the dc_i set at application level. The brake force control application after replication is shown Figure 2.6. The values of dc_i is set as $2, 2, 3$ and 2 for jobs j_1, j_2, j_3 and j_4 respectively. The replicas are represented as j_{1a}, j_{1b} for j_1 and similarly for other jobs. As mentioned the influence values between replicas are assumed to be 0 which is shown in the figure for example the value between j_{1a} and j_{1b}.

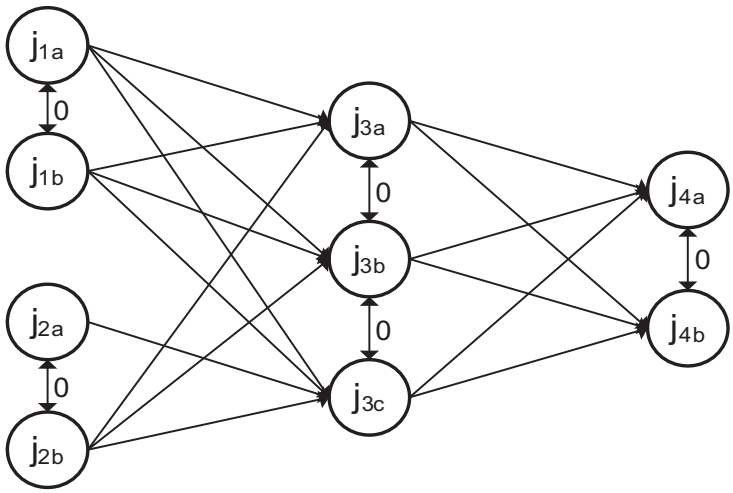

Figure 2.6: Brake force control application after replication

2.3.5 The Fault-Model

We consider both SW and HW faults, therefore a fault can occur in any job, HW node or communication link. The consequence of a fault is an error which can propagate from a source module to a target module explicitly

via an erroneous message sent by a faulty job or via some shared resources (implicit propagation channel). The failure of a job is defined as the violation of the specification in either the time or the value domain. A failure in the value domain implies that the content of a message does not conform to its specification. A timing failure implies that the send or receive instant of the message is incorrect (either early or late).

For *SW faults*, each job is considered as a unit for SW fault containment. A SW *fault containment module* (FCM) [79], should ideally by design, also ensure that errors do not propagate but are contained and tolerated. In case of SW faults, jobs are assumed to fail independently. Fault occurrence is considered within an FCM and over communication across them. The consequence of a fault is an error which can propagate from the source to a target by an erroneous message of a faulty job. SW faults occurring in one job propagate to another job if and only if there is a communication between them.

For *HW faults*, a node is referred to as a HW FCM. A HW node may contain resources shared across jobs, e.g., processor, memory and power supply. A single (*transient or permanent* [3]) fault impacting any of these shared resources is likely to affect several or all of the jobs running on the node. Therefore the shared resources introduce new paths for error propagation. Hence, in case of HW node failures (e.g., transient fault or permanent/crash fault) replicated jobs should be allocated onto different HW nodes in order to ensure FT of application services. Transient faults are the faults which appears for a short duration in the system and can affect the system for the short time period only whereas permanent faults cause a long term or permanent damage of a particular component or the overall system in the worst case. In the latter case the component needs to be repaired or be replaced with the backup one. In both cases FT mechanisms need to be provided. In the case of communication link, only transient faults are considered, e.g., in automotive cars electro magnetic interferences (EMI) can causes communication failure. We assume that whenever an EMI occurs, the communications on the bus is perturbed during a certain time period (transient errors in nature) and bits transmitted during this period is corrupted with a certain probability.

2.3.6 Functional and Extra-Functional Constraints

Embedded systems are heterogeneous (comprises of SW-C, HW-C and external resources) in nature which often operate with limited physical resources and interact with the environment and have to operate correctly in any circumstances entailing the consideration of a large set of constraints. The

co-design of such systems is therefore subject to constraints. For better SW-HW trade-offs both the functional and extra-functional constraints need to be considered at the early phase of the co-design process. Formally, the constraints are defined as the conditions that limit the possible designs from the dependability, timing or resource perspectives. In other words constraints are conditions that the system has to satisfy to ensure correct behavior (both functional and extra-functional). In general, they are divided into two categories: *hard* constraints and *soft* constraints. We define *hard* constraints h_C that need to be satisfied else the solution is not valid. For example, a *hard RT* system must respect all timing constraints. In order to be able to create a feasible solution, these constraints have to be satisfied during the allocation phase of the approach. *Soft* constraints are those whose violation makes the solution inefficient (a performance degradation from RT view) but not completely invalid. Often design objectives are formulated as soft constraints.

A set of constraints need to be extracted and defined for a particular embedded system design. We define them for designing dependable RT embedded systems. Some constraints can be derived from the functional and architectural specification and some others can be derived according to the extra-functional properties of the target system applications. Unlike single-constraint driven integration, e.g, as mentioned in the related work mostly satisfies only timing constraints, our approach is characterized by a concurrent set of constraints. Several constraints, which restrict the possible designs, are considered in our approach described as follows.

Binding Constraints

Binding constraint is described as the need of special resources by some particular functionality which entails a phenomena of one-to-one assignment of function and node. Some functionalities/jobs can only be mapped on a subset of available nodes due to the need of certain resources such as need of sensors, actuators or any dedicated HW like watchdog, DSP or ASIC.

Dependability Constraints

We first consider dependability as constraints and then as an optimization variable. Therefore dependability is taken directly in the process to achieve a required level by design. Primarily we define two aspects and formulate them as constraints as follows. Violation of these constraints may lead to a catastrophic consequences in the system.

Separation of replicas: This constraint is necessitated by the SC

jobs. Replicated jobs (jobs are replicated according to their degree of criticality) must be in partitions of different HW nodes in order to provide a certain level of FT.

SC and non-SC partitioning: We consider this in order to maintain strong partitioning between components of different criticality.

Timeliness Constraints

A hard RT system must respect all timing constraints. When these constraints are violated, a major malfunction may occur specially often in the case of SC embedded systems. Deterministic and predictable timing behavior is the primary concern for such systems. The timing constraints are often can be modeled as inequality constraints [68]. At design time, satisfaction of the following temporal constraints define a schedulable allocation according to deadlines.

Precedence relations: This constraint is necessitated by the communicating jobs where jobs are dependent with each other. When a job depends on the result of another job a precedence relation is established. This relation can be formulated as an inequality constraint as follows: $EST_i + CT_i \leq EST_j$. Where, job j depends on job i meaning that job j starts its execution only when job i finishes its execution.

Deadlines: Each application should provide services within a certain amount of time usually according to a respond to the environment. Therefore, application jobs should finish execution before their deadlines. This can be expressed by the following relation: $\forall_i\ EST_i + CT_i \leq D_i$. The deadline of a particular job must be greater than the sum of earlier start time and computation time of that job.

Computing Constraints

These constraints are extracted from the limitation of the processor capacity and are categorized into following two classes.

Computational capability: Each job runs on a processor with a certain amount of computation time. The sum of computation times of all jobs running on the same processor must be less than the computation capability provided by that processor.

Memory consumption: A job assigned on a processor consumes a certain amount of memory. The memory usage by the total jobs

allocated onto a processor cannot exceed the available memory capacity of that particular processor provided that multiple processors are not sharing a common memory device.

Communication Constraints

Following two requirements are set under this constraint.

Bandwidth requirement: This requirement is necessitated by the inter job communication. A sufficient bandwidth for communicating jobs on different nodes must be provided by the network.

Allocation/Sharing protocol: Communication between jobs and the required bandwidth depends on the considered protocol.

Additional Constraints

We define additional constraints as the constraints which are not directly addressed in the approach.

Physical constraints: The typical example of this constraint is the size and weight of all the materials and components used in a system. In automotive industry, even though mechanical parts are replacing by the SW components, the weight of the car is growing due to installing large number of electronic devices and wiring. The cabling and wiring used in vehicles and the number of nodes impact on the size, weight and space of the vehicle. Each node (ECU) in a car consume a certain amount of resource weight itself by using the micro-controller, boxing, sensors, actuators and other electronics. The weight can be reduced by using new and lighter material, however, the mapping decision does not impact on this. The mapping can impact on the weight of the wiring and the number of nodes used. This can be benefited from our targeted integrated design approach by reducing the number of ECUs and wiring. Reducing the weight has a great effect on reducing the fuel consumption of the car.

Power dissipation: Recently power management is becoming one of the key issues for designing embedded systems. Specially reduction of power consumption is very important for systems which carry their own energy sources. It would be a critical factor for the recently developed electric cars (can drive 100km with a 6 hours of battery charge) which are only battery operated. A power aware design is preferable than

simply a low power design for such systems. Therefore it is necessary to integrate efficient power management techniques to the system level co-design process. A system level power aware scheduling technique is derived in [110] for Mars Pathfinder. Power can be modeled as a pure constraint as well. A job runs on a processor consume a certain amount of power. The total power consumption by a mapping must not exceed a maximum power limit given by the system. Nevertheless, we have considered power as a system level optimization variable with an extension to our approach (see Section 8.3 for more detail on modeling the power).

Chapter 3

System Level Co-Design and Optimization Approach

The design methodology of embedded systems is an evolving process. Also the growth in the complexity of the design is a key issue as functionalities and extra-functionalities get added leading to the need for new design methods and techniques. The system level co-design process enables better functions (SW) and architecture (HW) trade-offs analysis while dealing with extra-functional attributes which have strong consequences on design objectives. Consequently, we develop an extra-functionalities driven system level co-design and optimization methodology for designing an integrated embedded architecture (IEA). Our proposed SW-HW co-design methodology focuses on design analysis, mapping, optimization, prototyping and their evaluation/validation at system level of abstraction. A key issue of the methodology is that we provide quantification of various extra-functional attributes such as FT, influence, schedulability, resource utilization and power.

Before we start presenting the system design and optimization flow, we describe some increasingly important design aspects of dependable RT embedded systems which are later then taken into consideration in the co-design process. An important issue of system design is the co-design space exploration which helps to traverse the overall design space is explained next. In order to efficiently explore the search spaces, various techniques and the way of tackling the constraints are described. We mathematically formulate the problem of exploring the design space by resource allocation and optimization. The mapping problem is then formulated as a constraints satisfaction as well as a multi objective optimization problem. We develop a generic framework in order to perform the system level co-design and optimization for the design of an integrated dependable RT embedded system. The framework mainly comprises the following design issues: requirements analysis and

specification modeling, SW-HW mapping, design optimization and prototyping. This chapter provides the answers of the part of the *Research Questions 1, 2, & 3* [RQ1], [RQ2] and [RQ3] which are related to design constraints, optimization and the co-design framework.

3.1 Embedded Systems Co-Design Criteria

There may be several ways in which a co-design methodology can be developed, thereof, we need to define what constitutes a good design and what factors need to be considered that impact on the design and its optimization. The consideration of these design issues strongly depends on objectives of the target system design and on the considered system model. Our goal is to design dependable RT embedded systems thus we consider criteria from those perspectives. In order to be able to create a feasible design the first criteria is to satisfy the hard constraints and then meet the desired system requirements and objectives for an optimized system design. These are described in the following sections.

3.1.1 Satisfaction of Constraints

We refer this design issue as satisfaction of hard constraints, which implies the absolute constraints on behavior, whether semantic, temporal, or others. The constraints for example separation over replication to provide FT and precedence and deadline constraints to ensure RT properties are hard constraints. These constraints must be satisfied to create a feasible design. However it is not considered how the jobs and replicas are assigned to achieve better dependability and how the schedulability can be optimized, e.g., reducing the length of the schedule. These are the measures to optimize and to assess the design. While some constraints can be evaluated a priori others can only be checked after assignment, if so, this is always the primary concern. The constraints defined in the previous chapter are satisfied during the allocation phase of the design process.

3.1.2 Dependability

Since dependability is the primary objective for the design of SC systems, as mentioned earlier our aim is to achieve and optimize dependability by design. In this regard, we first replicate the high criticality jobs and then we endeavor to minimize the *influences* between jobs as well as nodes in order to increase the fault containment. There is a probability of error propagation

from one node to the other due to interacting jobs allocated in different nodes. Traditionally system designer design the system just to provide FT. However the system should not only provide dependability for critical applications, we should also design the system as good as possible from dependability point of view, e.g., minimizing influences in order to reduce the error propagation probabilities to provide better FT. Replicated jobs must be assigned onto distinct HW nodes, however, it is necessary to find the criteria in which we can assign them (replicas) in a way such that dependability is optimized. Given the first priority to dependability, the properties from this perspective are not compromised while generating the design. Thus we consider design optimization variables from dependability perspective.

3.1.3 Real-Time

A hard RT system must respect all timing constraints, e.g, jobs have to finish their execution by a certain amount of bounded time, i.e., by their deadlines. However in case of soft RT, this condition can be relaxed. It is acceptable for soft RT jobs to run in a performance degradation mode. In classical scheduling theory, in order to evaluate (performance measure or optimality criteria) scheduling results typically usage variables are for example minimizing the sum of completion time, minimizing schedule length, minimizing the weighted sum of completion times, minimizing the no of processors required, or minimizing the maximum lateness [111]. Some of these variables for example sum of completion time may not directly satisfy the timing properties. Even it may not cover several aspects like deadline or periods while optimizing a design using only a single variable. *Slack* is also a valuable RT properties which is utilized to optimize different variables. It is defined as the gap remain between two jobs execution and how unloaded a processor is. The aim of considering variables from this criteria is to proper utilization of the computing power and to achieve better performance of the design.

3.1.4 Resource and Power Consumption

Resource utilization is a classical factor for optimizing the design of embedded systems. Resources can be utilized from different point of views, e.g., utilizing the bandwidth resource, utilizing the number of nodes, utilizing *load balancing* and *power consumption*. Load balancing is the technique where jobs/processes are evenly distributed among the nodes to leave as much slack as possible in the design. Resources are utilized either from performance or cost point of view. For instance, if less number of nodes is required in the design that means it requires less amount of cost. Now a days a significant

focus is being given on the power consumption issues at design time due to the increasing fuel prices.

3.2 Co-Design Space Exploration

Co-design space exploration (CSE) is a process of investigating different mapping solutions in the global design space in quest of identifying the most prominent solution, i.e., an optimal solution. CSE enables the system designer to germinate and compare various design alternatives under specified constraints and variables in order to find out an appropriate either a feasible or an approximated optimized solution. The exploration techniques tries to find a feasible solution from an exhaustive of K^Q (allocation of Q jobs onto K nodes) ways of assignment from the design space. Adding different objectives into the design process makes the problem more complex. Thus the satisfaction of multiple concurrent constraints and involvement of enormous trade-offs between diverse competitive and conflicting design criteria/variables make the CSE difficult. Strategies and methods are needed for efficient CSE which can reduce the design space as well as can cover the overall design space.

In order to cope with the increasing design complexity, our system level co-design allows the system designer to explore the search space at early phase of the design. There are various methods and techniques, heuristics and hyper-/meta-heuristics one can apply to explore the design space [61; 98; 112; 52]. The search spaces is guided/pruned by the satisfaction of different constraints. We apply constructive heuristics to prune the search spaces and use metaheuristic technique to cover the design space. Finding a feasible solution in the design space is comparable to a resource allocation problem on the other hand investigating different better quality solutions is compared with the optimization problem. In the next section we discuss different techniques of how to handle the constraints in order to be able to reduce the design search spaces. We then provide a mathematical formulation of the mapping and its optimization problem.

3.3 Constraints Handling

The precise definition of requirements reduces the design space as they define constraints and objectives in addition to the designated functions, that limit the possible mappings from the dependability, temporal or resource perspectives. It helps to avoid the exploration of infeasible design alternatives [43].

Moreover a properly selected objective function may control an automated synthesis process to deliver (sub)optimal solutions. Applying the efficient search methods and techniques for constraints satisfaction, the unnecessary exploration of infeasible regions can be avoided and can effectively guide the co-design space exploration. In the following we discuss how the constraints are handled during the mapping and describe different search techniques employed in the allocation phase of the design process. The constraints handling techniques are employed and satisfied during the mapping algorithm.

Constraints Prioritization: The efficient management of a large set of constraints can rely on *constraints prioritization* [78]. The system designer assigns priority levels to the individual constraints estimating their order to guide the searching process. This technique can be used for partitioning complex constraint systems into sequentially solvable blocks by exposing causal interdependencies between individual constraints. For instance, the replication of SC jobs precedes in a SC application all other decisions on job allocation (as is expressed by the topmost priority assigned to the related constraints).

While searching for a feasible mapping the constraints are checked in a priority basis to satisfy them on each time node assignment, i.e., an evaluation is performed for the assignment in order to increase the search efficiency. There are several search techniques, e.g., *consistency enforcing, retrospective, backtracking*, one can apply a priori by which the design space can be explored in a better and efficient way. The backtracking is seldom needed when search procedure integrates the mentioned techniques (*prioritization and a prior* check) together with the *ordering heuristics* presented in Section 4.3. The backtracking is applied only when there is an inconsistency, i.e., a dead-end is reached.

Consistency Enforcing: *Consistency enforcing* prune the search space by avoiding the local inconsistencies in the assignment process [50]. This technique allows us not to try to assign the replicas on the same node and is also enforced by the FT constraint. The assignment process does not need to know *a priori* (before checking the constraints) that whether there is a replica already assigned. When a replicated jobs are instantiated to be assigned from the list, a new node is selected for it to be assigned. Once a job is selected from the ordered list (the jobs and nodes are ordered), this process tries it to assign onto a node satisfying all the constraints without backtracking. *Look-ahead* method can also be applied together with the consistency enforcing.

Look Ahead Technique: The *look ahead technique* or the retrospective technique is characterized by the assignment of a job to a node while checking other jobs that are already assigned in this node in order to avoid conflicts. For example when checking the timing constraints for a job it is necessary to check the schedulability combining with the jobs which are already assigned onto a particular node (all the possible combinations of jobs are checked). Ordering jobs and nodes is the first step of look ahead technique meaning that we know in advance which job to be assigned next.

Backtracking: This mechanism enables us to undo some previous assignments in case there is an inconsistency[1], i.e., no feasible assignment is possible with the current search path. The backtrack process goes back to the earlier assignments and changes them to the alternative feasible ones. One simple and easy backtracking mechanism is the chronological backtracking which systematically changes the most recent assignment and tries alternative ones. If it is not possible then it respectively goes back to the next most recent assignment. We have implemented this technique which includes moves like *relocate* (relocating a job to a different node) and *swap* (swapping the nodes between two jobs). If there is no assignments left to undo, i.e., search reaches its initial state, then the mapping is infeasible and the process terminates.

3.4 Mathematical Formulation of the Problem

The crucial problem of the system level co-design process is to map the jobs onto suitable nodes in an optimized way. The problem is divided into two parts: creating a feasible mapping and finding an optimized mapping. We formulate the creation of a feasible mapping as a generalized resource allocation problem which is modeled as a *constraints satisfaction problem* (CSP) [50]. The problem is characterized by a given set of jobs $\mathcal{J} = \{j_1, ..., j_m\}$, a distributed computing architecture associated with k nodes $\mathcal{N} = \{n_1, ..., n_k\}$ and by a set of constraints $\mathcal{C} = \{c_1, ..., c_l\}$. A solution to this problem is an assignment of each of the n jobs to one of the k nodes such that all constraints $\mathcal{C} = \{c_1, ..., c_l\}$ are satisfied and objectives are met. Once the objectives (either single or multiple) are met the design is said to be optimized. The set of all possible mappings for a given set of jobs and nodes is called the *design space (X)* that includes feasible (X') and infeasible region ($X - X'$). In order to find a feasible solution the overall design space

[1] An assignment when it violates any constraints is said to be inconsistent.

X is explored. The constraint surface (see Figure 3.1) divides the design space into two regions: *feasible* and *infeasible*. Constraints that represent limitations on the behavior or performance of the system are termed as *behavior constraints* (e.g., FT and RT constraints) and that represent physical limitations are called *geometric/side constraints* (e.g., weight, size, binding constraints) [113].

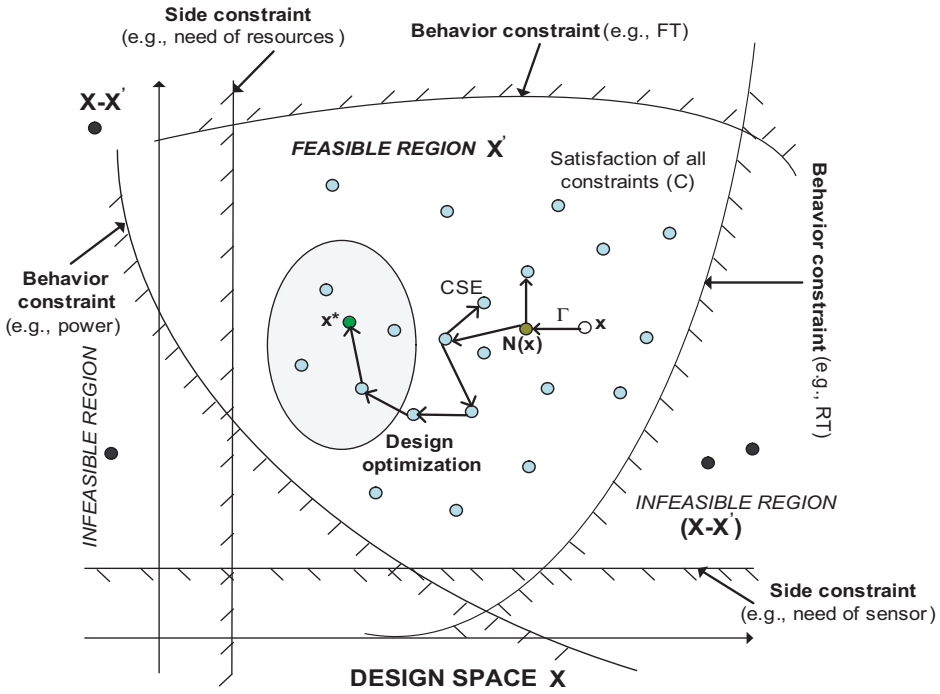

Figure 3.1: Hypothetical design space

A *hypothetical design space* is shown in Figure 3.1, where infeasible regions are indicated by the hatched line. A *point* x in the design space X represents a mapping of jobs onto nodes. Points located in the region of constraints satisfaction are feasible points. A mapping is either feasible/aceptable or infeasible/unacceptable. A *feasible mapping* is a solution which satisfies all the constraints $\mathcal{C}$. If a constraint is not satisfied then the mapping is *infeasible*. These constraints are referred to hard constraints. The *neighborhood* space $N(x) \subseteq X$ of a point x is the set of all points that are reachable by performing a *move* operation (e.g., relocating a job to a different node). We employ a *transformation operator* (Γ) to perform the move operation (see Section 5.3 for details), i.e., to explore the neighborhoods leading to covering the design space. This parameter is used either creating an initial feasible mapping when backtrack is necessary or finding an optimized mapping both from fea-

sible and infeasible points. Our mapping algorithm explores the global space X in order to find a solution in the region of $X^{'}$. It is a *constructive heuristic* which creates a feasible mapping for a set of jobs and nodes in every single run of the algorithm.

Usually there exist many mappings that satisfy the defined constraints, i.e, there are more than one feasible mapping. Therefore measures are needed to find a suitable and as good as possible mapping. The *value* of a point is a measure of the suitability of the mapping represented by that point. The function $f(x)$ is used to measure the value of a point in the design space. We term this function as MVO function $MVO(v)$ (see Section 5.3.1 for details), where v is the variable that needs to be quantified and optimized. A very important aspect to determine the value of this function is to quantify the considered design variables (see Section 5.2 for details). For an optimization problem, which minimizes the value of objectives/variables, good mappings have low values. The task is to find a mapping $x^* \in X$ with the lowest function value, i.e., $f(x^*) \leq f(x)\ \forall x \in X$. x^* is the optimized mapping from the design space X. However guidance of heuristics is necessary for an efficient search in the global design space in order to obtain an optimized mapping with less computation cost. In order to prove this we have performed a comparative study with [42], where scheduling (ordering jobs execution) is implemented and conducted in a CPLEX based tool [114]. CPLEX is an ILOG software product for solving linear and mixed integer programming problems. As mentioned, we first create a feasible mapping and then this feasible mapping is provided as an input to the optimization algorithm and feasibility is maintained throughout the quest by an external function call (see Section 5.3.3 for details), therefore the problem space remains in the feasible region $X^{'} \in X$ (set of all admissible solutions), which reduces the search space considerably. We strive to find the optimized design $x^* \in X^{'}$, where $f(x^*) \leq f(x)\ \forall x \in X^{'}$.

For an MVO problem, the value function is structured with multiple variables, which evaluate or measure the mapping. The function is represented as an optimization function in additive form (see Section 5.3.1 for details) as follows. The constraints are not formulated in this function (see Equation 3.1). They are satisfied separately by the implementation of an external function. If both constraints and variables are included in a single value function it would be difficult to poise these two factors together, e.g., setting the trade-off factors. On the other hand if the number of variables and constraints are increased in a single function then this increases the complexity of the problem. Thus we do not consider the constraints satisfaction function with the value (optimization) function. Note that for a single variable

optimization problem the value function is straightforward.

$$min\ f(x) \equiv MVO(v) = \sum_{i=1}^{n} \lambda_i f_i(x_i)$$
$$= \lambda_1 * f_1(x_1) + \lambda_2 * f_2(x_2) + ...\lambda_n * f_n(x_n) \quad (3.1)$$

where, λ_i are the trade-off factors and n is the number of variables. The value of each variable is represented in a vector form in a matrix $M[v]$ for a particular mapping x. In this case the aim is to find the best non-dominated set of variables, i.e., a set of pareto solutions that satisfy the system requirements and objectives.

$$M[v] \equiv \forall i M[f_i(x_i)] \equiv M[v_1, v_2, v_3, ..., v_n] \quad (3.2)$$

where, $x_1, x_2, ..., x_n$ are the design points, $f_1(x_1), f_2(x_2), ..., f_n(x_n)$ are the corresponding value functions of the variables and $v_i = f_i(x_i)$ for a specific mapping x.

It is already mentioned that the mapping problem is NP hard, therefore, in order to reduce the complexity we divide the process into subproblems. The mapping itself is divided into two subproblems: allocation and scheduling. First, we create a feasible allocation by using the proposed algorithm satisfying all the defined constraints including schedulability. The algorithm considers the proposed jobs and nodes ordering heuristics in its construction. The jobs and nodes are ordered before the allocation takes place which assist to find a feasible solution with less number of iterations (see experimental results in Section 6). During the allocation phase all the constraints are satisfied in a prioritized manner in order to be able to create a feasible mapping. If a constraint is not satisfied in *Step* 7 of the Algorithm 3 we perform backtracking in *Step* 8 (see Section 4.4). A validation test for the allocation is then performed so that it can be scheduled. The output of the algorithm derives the basic scheduling. An optimized solution can then be easily found by using CPLEX or any other approaches presented in Section 5. The initial feasible mapping guides the optimization process in an efficient way to find the solution (see the validation and comparative study in Section 6.4).

3.5 The Integrated Design Framework

The proposed system level co-design and optimization methodology for the design of integrated dependable RT embedded systems meticulously guides the multiple constraints and variables driven mapping towards an optimized mapping. We first perform a heuristics based systematic mapping of SW-Cs

onto HW-Cs such that constraints are satisfied. We then provide a multi objective optimization technique called as MVO to quantify and model the variables and to optimize the design. The variables are considered from dependability, RT, power and resource perspectives. The design is then prototyped in order to deploy the system onto a target architecture. The overall co-design optimization steps and the flow are depicted in Algorithm 1 and in Figure 3.2 respectively. In the following section, we simultaneously describe all these necessary steps and the design flow.

3.5.1 The Overall Co-Design Methodology

The co-design process starts with the idea of separation of concern at system level by characterizing the SW and HW model separately. The models are specified from the description of system functional, performance and dependability requirements. At first the requirements need to be elicited. The requirements can be captured for example by using the technique such as [115], which collect both functional and extra-functional user requirements specially giving the focus on describing the *context-driven functionalities*. Context-driven functionalities describe the special types of functions in an application including extra-functionality. The system requirements are abstractly modeled in the SW model without the knowledge of the HW architecture. The co-design process continues over setting the architecture and HW platform resources and performing the mapping, optimization and the evaluation. We assume that the description of the SW model and the candidate set of HW resources and services are available prior a mapping can take place. Essentially, the SW model is characterized with jobs and their properties like functionality, computation time, degree of criticality, which are extracted from the system requirements and specification document. This constitutes the system architecture and SW model in *Step 1* (shown both in Algorithm 1 and in Figure 3.2).

In *Step 2*, constraints are modeled, which need to be satisfied during the mapping to ensure correct system behavior. All constraints imposed on application or platform level are extracted from the specification or defined by a system designer before resource allocation can take place. Prior to do the allocation job information is needed as well, which includes measurements or estimations of job code and data size and timing requirements (as shown in Figure 3.2). Measurements on existing code or estimations can be used to obtain timing information, e.g, determining jobs CTs. For SC jobs, designer has to specify the required degree of *replication* in order to ensure FT. Other types of constraints, such as the computational capability and memory capacity of the computing nodes as well as the network bandwidth have to be

Algorithm 1 Generic methodology for system level design optimization

1: Derive the system architecture and model;
2: Extract design constraints;
3: Define design variables;
4: Order jobs and nodes;
5: Generate an initial current mapping - apply heuristics;
6: Generate candidate mapping - exploring neighborhoods;
7: a) Compare candidate mapping with the current mapping;
b) Go back to *Step* 6 until stoping criteria is met;
8: Define minimum requirements to select the mapping (the *aspiration values*);
9: Assess/evaluate the mapping and return the good mapping (a *near-optimal* one);

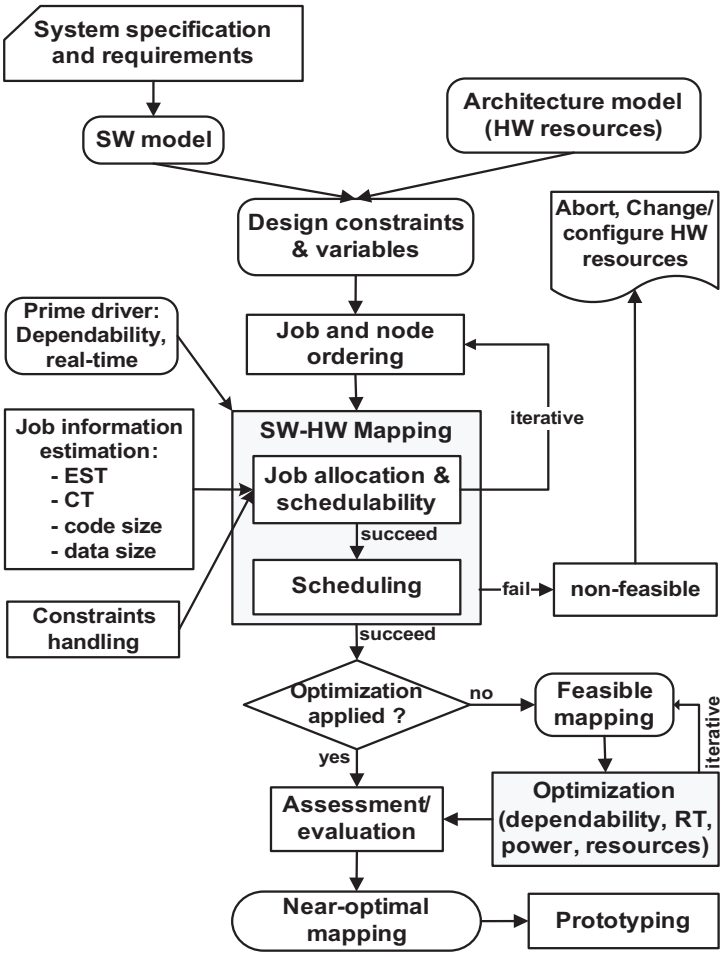

Figure 3.2: System level co-design and optimization flow

extracted from the platform details. In the previous chapter we have already defined a set of such constraints. Design variables are defined in *Step 3*,

which are employed in the *mapping optimization* phase shown in Figure 3.2. Variables are used for capturing the design criteria and they strongly depend on the objectives of the system design [116] and on the considered system model.

Mapping algorithms need heuristics to efficiently prune the search space and to achieve good performance. Of particular importance are job and node ordering that decide which job to assign next and what node to assign that job onto. Job and node ordering are described in *Step 4*. A crucial and the most important part that arises at this design stage is the mapping of jobs onto suitable nodes. An initial mapping is created in *Step 5* where allocation and scheduling is performed off-line in the early design phase. It is necessary to satisfy the hard constraints during the creation of an initial mapping. The allocation is performed iteratively and the outcome is used for scheduling. If allocation or scheduling fails, e.g., if all constraints are not satisfied due to the lack of enough resources then an infeasible mapping is aborted (see Figure 3.2). Furthermore we may need to change or reconfigure the architecture with more resources, e.g., adding more nodes to the architecture or adding certain sensors/actuators to the nodes in order to create a feasible mapping. The result of this step is a feasible mapping and is likely to be insufficient from a system design perspective. However use of this mapping as an input to the optimization resulted in a significant performance gain and reduces the search space considerably. We elaborate more on this in the evaluation chapter (see Chapter 6). The purpose of the rest of the steps is to find a better mapping by using the developed quantification and optimization techniques. In this regard, *Step 6* generates a candidate mapping from a set of possible solutions. Candidate mapping is generated by exploring the neighborhood spaces. In order to select a better design, the candidate mapping is compared with the current mapping in *Step 7*. The value of the mapping is calculated by using the MVO function. If a better mapping is found, the current mapping is updated. This step is iterative so that the comparison can be made with all the possible mappings. This way the co-design space is explored. The optimization aspect is shown at the end part of Figure 3.2. In *Step 9*, the mapping is evaluated and assessed to ensure that it satisfies the minimum system requirements defined in *Step 8*. Essentially, we are interested in finding a *near-optimal* mapping meeting particularly the design objectives of dependability/FT, RT and resource utilization.

Overall, the integrated design framework encompasses the following design challenges which are further elaborated in the subsequent sections and chapters:

- Requirements analysis,

- SW-HW mapping – allocation and scheduling,
- Design optimization, and
- Prototyping

3.5.2 Requirements Analysis and Specification

At the very beginning of the design process the system designer has to specify the functionality and extra-functionality of the target system domain. Specifying systems in a proper way is an important task. A system may fail either because it does not act in accordance with the specification, or because the specification did not describe its function adequately [28]. For example, defining the degree of criticality for SC systems, the criticality has to be set properly because the FT design decision depends on it. Therefore a good and error free system design relies on adequate requirements specification. If the requirements are inconsistent, incomplete or invalid then the design will probably be inappropriate or even useless. Moreover faults/errors detected during development and testing phases are one order of magnitude more costly to fix them those detected during requirements phase [117]. Thus a consistent and complete specification is essential.

The consideration of varied mixed criticality heterogeneous systems requires advanced techniques to ensure properties such as correct behavior and adequate performance. These difficulties yield complex requirements specification and raise many challenges along the way. As SW is an important factor for innovation and creation of value in modern industries (SW will make an estimated 38% value creation of total production costs in the year 2010 [118]), high quality and reliability SW should be achieved. One critical precondition for reaching high-quality automotive SW is correct SW requirements. The development of embedded SW systems is becoming a cause of widespread concern due to the risk of errors. Specification errors represent the majority of the failure caused in SW development. Therefore an error free requirements specification is the first design goal to achieve dependability by design. Once exact and correct requirements are elicited, different specification representation models and languages can be used to involve them further in the co-design process. Examples include sate oriented models [119], which represent the function of a job with a set of states and a set of transitions between them, e.g., petri net, finite state machine etc. [8] – page 33 and modeling approaches such as unified modeling languages (UML) [120], Y chart modeling language [62] and the SAE (society for automotive engineers) architecture analysis and design language (AADL)[121] which is a part of the architecture description language (ADL) [122].

3.5.3 SW-HW Mapping

Proper execution of the specification describing jobs functionalities onto node processors entails the decision of how and which nodes will host what functionalities while maintaining large variety of design constraints. This process is carried out by performing the mapping of SW (job functionalities) onto HW (nodes). The mapping is a crucial design step of the co-design process enabling trade-off analysis between SW and HW. Considering a wide range of constraints such as dependability, timing and resource constraints, Chapter 4 presents a heuristic based systematic resource allocation approach for the consolidated mapping of SC and non-SC applications onto a shared distributed architecture such that their operational delineation is maintained over integration. We describe different mapping strategies and policies prior to do the assignment of jobs onto nodes. The strategies include FT, reduction of communication and influence, schedulability and resource utilization. These policies are taken into account while constructing the mapping. We propose an ordering heuristics to prune the search space. Thus a mapping is created *efficiently* by reducing the search space and finding a feasible solution with less number of iterations. The premises of the ordering heuristics is to order the jobs and nodes according to their importance to take part in the assignment process. Our mapping algorithm is a multi-phase assignment process that explores various combinations of mappings subject to specified constraints. The algorithm considers different criticality jobs at different phases in order to achieve a maximum *separation* (minimum influence) between them.

3.5.4 System Design Optimization

The design should make the best use of available resources. The mapping of jobs onto nodes should be as effective and efficient as possible necessitating the design optimization. Optimization looks for the best possible solution in a design space. The optimization process improves the system design either from single objective or from multiple objectives. The example of optimization can be found in everyday real life such as finding the shortest path between home and office to save time and fuel cost, getting the suitable warm water by tuning the hot and cold tab at the expense of less hot water, turn on the oven *a prior* to cook to save energy and cooking time etc. The design of heterogeneous embedded systems comprises of SW and HW together with the consideration of various emerging design criteria from such systems entails performing the multi objectives optimization in the co-design process. Moreover in order to avoid inefficient resource usage, optimization is applied

since faster processors and large number of nodes and resources as many as needed may not be affordable due to the cost constraints. As our provision is to design an integrated approach where optimization allows transferability of functions from heavily used nodes to a less used node making the resources properly utilized. New and innovative functionalities may need to be added and existing functions may need to be upgraded over system life time. Optimization supports such scenario where new functionalities to be integrated later in the design as it enables design adaptability and scalability [123].

In our design process during the mapping stage we first allocate jobs onto nodes according to the heuristic of reducing influences so that the mapping algorithm can possibly find a better solution. However there can be several variables that need to be improved simultaneously and can have more than one feasible mapping and even more than one local optimal or near-optimal mapping. Therefore we need mechanisms to explore all the feasible regions and consideration of different variables in order to create a suitable optimized mapping. Consideration of multiple variables together entails the trade-off analysis between variables that can be resolved by applying the weight according to the importance of the variable as well as the system designer can informally weigh the trade-off. Different objectives/variables are combined into a single function by applying weights expressing their importance, and/or reached by using multi objective optimization techniques presented in Chapter 5. In order to find a near-optimal solution a *Multi Variable Optimization* (MVO) approach is developed. The approach considers satisfaction of multiple constraints and optimization of several competing variables at the same time. Each of the considered variables are estimated and modeled through appropriate technique and optimized during the MVO process. The design is optimized by using an existing metaheuristics based optimization algorithm. We use simulated annealing (SA). Section 5.3 depicts the reasoning for choosing SA and elaborates the MVO-SA approach.

3.5.5 Prototyping

After the integration and evaluation process once the system designer is satisfied with the developed design, the design is prototyped in order to deploy a system onto the target architecture. In order to ease the development process a set of supporting tools is integrated in a single platform and the design is prototyped. For this we have developed a transformation based design approach adhering to model driven development [124] and platform based design [125] principles. By the guidance of the system level co-design and mapping process, we generate platform specific post integration model which controls the deployment task of the target applications. The prototyp-

ing starts with specifying the systems in an abstract model we call it platform independent specification model. Primarily, in the prototype UML [120] and XMI (XML (extensive markup language) Metadata Interchange) [126] are used as modeling languages for the specification and representation of different models. We elaborate on prototyping in Chapter 7.

Chapter 4

Dependability Driven SW-HW Mapping

Embedded systems functionalities are increasingly being implemented in SW. However the availability of physical resources is not necessarily such that each SW-C can be allocated to its own HW node. The situation is limited by physical (space, size), weight and economic constraints. Therefore mapping of those SW-Cs needs to be performed onto limited and shared HW resources. As already mentioned, the mapping problem becomes intractable to find a feasible solution and invariably leads to a constraints satisfaction problem. It becomes even more complex when considering extra-functional requirements and objectives in the design. Thus, in general this problem requires development of heuristic based techniques to solve them in a tractable manner. Given the highest priority on designing SC embedded systems, a heuristic based dependability driven mapping of SW functionalities onto HW nodes is what is needed while satisfying extra-functionalities and HW imposed constraints.

We develop a framework which systematically guides such mapping of SW-Cs (jobs) onto HW-Cs (nodes). The main characteristics of the mapping are *(a)* to provide FT assuring a certain level of dependability desired by the user, *(b)* to enhance dependability by reducing the probability of error propagation, and *(c)* to satisfy the timeliness properties (RT) through schedulability analysis. Other requirements and constraints, e.g., satisfaction of need of certain resources and desire to reduce the communication load are taken into account as well to ensure a valid suitable mapping. A key feature of the mapping is the job and node ordering heuristics. During the assignment process jobs and nodes are selected from an ordered list. The mapping is based on these premises which are discussed in details. This chapter mostly covers the answer of *Research Question 3* [RQ3], particularly

the creation of a feasible SW-HW mapping. The developed algorithm for doing the mapping employs *ordering heuristics* (presented in Section 4.3) and different mapping strategies in its construction. The constraints set $\mathcal{C}$ are satisfied during the assignment in each phase of the algorithm. The creation of a feasible mapping is accomplished by using an iterative process. In order to facilitate the representation of the mapping and ordering heuristics, a supporting data structure is described. To demonstrate the usefulness and effectiveness of the approach the mapping process is applied to an actual automotive system which is illustrated in Section 4.5. For this illustration we use a HW architecture where nodes consist of dual core processors. One core is for the assignment of SC applications and the other for non-SC applications. The experimental evaluation of the mapping is given in Chapter 6 and is prototyped in Chapter 7.

4.1 Basis of the Mapping

The strategies are outlined that drive the SW-HW mapping assuring the appropriateness of system design from FT and RT perspectives. We start by discussing the strategies for ensuring FT, followed by discussions on the desire to reduce sensitivity to errors by influence reduction. Different FT schemes are described which either tolerate permanent or transient faults or tolerate both types of faults. The schedulability analysis is then presented. In order to produce an schedulable mapping all these strategies are considered in the allocation phase of the design.

4.1.1 Fault Tolerance Schemes

Traditionally FT scheme predominantly utilized HW based redundancy, e.g., Multi-computer Architecture for Fault Tolerance (MAFT) [127], Maintainable Real-Time Systems (MARS) [128], XBW [4] and JAS 39 Gripen [129]. The active replication based FT is used in order to tolerate both permanent and transient faults. Usually multiple HW components/nodes are formed as a single unit called as *fault tolerant unit (FTU)* in order to tolerate either one permanent and/or one transient fault. When a node detects a fault, it falls silent and other replica nodes provide the necessary services. In this approach adding a new function requires to add a new HW node which is further needed to be replicated to provide FT. Hence this method of redundancy incurs high HW costs for adding new functionalities.

Thus, in distributed hard RT systems, FT is usually achieved through active SW or timing redundancy. In case of active replication, critical SW

components/jobs in the system are replicated and the replicas perform their services in parallel [130]. The technique employs replica deterministic agreement protocols, e.g., assure that all replicas start with the same initial state and perform the same operation. For timing redundancy, once there is a fault during the primary execution of a job it repeats the execution from the beginning. The FT scheme presented in this thesis ensures dependability through replication of jobs from SC applications. FT is provided by allocating replicas of jobs onto different nodes and either having recovery replicas to take over when a failure is detected, or use voting to mask the failure of a job. As jobs from an application may not be equally critical, all jobs from a single application do not need to be replicated to an equal level. The degree of replication of jobs is specified by the system designer based on the level of criticality, e.g., derived from the SIL level or from the specifications of the system or from the experimental vulnerability analysis [131] results. If the user sets a criticality degree (usually based on the knowledge and complexity of the application) uniformly on an application, all the jobs from that application have to be replicated equally. Replication of critical jobs makes the system more dependable. However overprotection leads to brute replication that may in turn come at the expense of increased hardware cost, power and schedulability. Thus a suitable degree of criticality needs to be set for each application jobs. We have also investigated different techniques complementing active SW based replication, such as re-execution, checkpointing [132] or roll-back recovery and interplay of these techniques (e.g., the interplay between active replication and re-execution [85]). Figure 4.1 shows the trade-off between different redundancy based FT techniques such as spatial and temporal redundancy, where the system tolerates 2-transient faults. Prior to execute any FT scheme, the fault needs to be detected. The fault detection process detects the existence of faults in the system either implemented with the FT schemes or implemented separately. Examples of fault detection techniques include signatures, HW watchdogs, assertions, comparators etc [133]. The overheads in time for fault detection and recovery always need to be considered with the computation time of particular job.

As we see from Figure 4.1 (a) that the SW based active replication uses more resources while taking less time to finish the computation. This configuration tolerates permanent faults as well. Whereas techniques like re-execution and checkpointing (and combination with replication) use comparatively less physical resources but incur large time overhead and are only applicable for transient faults (see Figure 4.1 (c), (d) and (e)). The primary backup technique computes a replica if there is a fault or error in the primary replica (the replicated job which starts executing first).

Under the assumption of transient fault, [85] has shown that the inter-

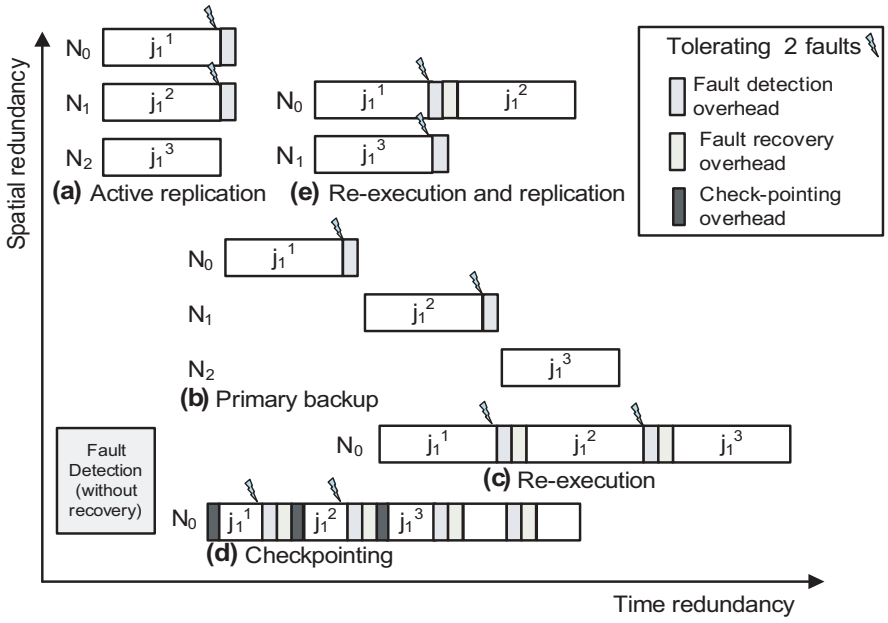

Figure 4.1: Trade-off between different FT schemes (a)–(e)

play of active replication and re-execution can provide an optimized system design from the scheduling point of view and thereof provide FT under limited resources. When the slack (defined as how unloaded a processor is) in scheduling does not allow the entire job re-execution, checkpoints may need to be inserted. The checkpointing or roll-back recovery requires less amount of computation time than complete re-execution on average. So the combination of active replication and checkpointing would be even a better technique than the combination of replication and re-execution for tolerating transient faults [134], which may tolerate more number of faults and may require less amount of computing resources per se. Thus a combination of replication and re-execution or checkpointing can be used as an FT scheme when transient faults are assumed as shown in Figure 4.1 (e). The desired FT technique depends on the considered fault model and also depends on particular application requirements. If an application needs to tolerate a permanent fault it has to be replicated and on the other hand if it needs to tolerate only transient faults then re-execution/checkpointing would suffice provided that the application meets the deadline.

In this work we consider SW based active replication for tolerating both transient and permanent faults. The reasons for choosing active job replication over roll-back recovery or checkpointing is that while on one hand we consider tolerating permanent faults and on the other hand in hard RT

systems roll-back recovery is often of limited use [43] – page 13, due to, e.g.:

(i) As the roll-back recovery can take an unpredictable amount of time, it is difficult to guarantee the deadline after the occurrence of a fault,

(ii) An irrevocable action which has been effected on the environment cannot be undone,

(iii) The temporal accuracy of the checkpoint data is invalidated by the time passed between the checkpoint time and the instant now and

(iv) Usually a perfect fault detection mechanism and permanent fault free data storage are assumed which may not be the case in practice.

In the following we briefly discuss several FT schemes which are shown in Figure 4.1.

FT Scheme 1: Active Replication

SW based active replication is used as an FT scheme which depends on the SW redundancy in spatial domain where replicas execute in parallel on distinct computing nodes. SW-Cs are replicated according to their degree of criticality. All the replicas execute their operations independent of fault occurrences. If there is a fault in one replica then other replicas provide the necessary services. This type of FT scheme is shown in Figure 4.1 (a) where the job tolerates either 2 permanent or transient faults. This FT scheme though take more resources, tolerates both permanent and transient faults. The possibility of meeting application deadline is higher as comparatively it takes less time than other FT schemes which are based on timing redundancy. As mentioned earlier with the reasons that we use active replication based FT scheme in our approach. For the sake of completeness we briefly discuss about the following schemes where most of them are based on timing redundancy.

FT Scheme 2: Primary Backup

In the primary backup FT scheme the main or the primary job run on a computing node and provide services until there is a fault. If there is a fault in that particular job then the backup replica starts execution on a different node and provide the necessary services. Again if there is a fault (in case it tolerates 2 transient faults) in this backup replica then the third replica takes over which is shown in Figure 4.1 (b). However this configuration can tolerate permanent faults as well. One replica executes at one time, when there is a fault the next backup replica is triggered to execute. An FT scheme by

using the primary backup scheme for an static scheduling of heterogeneous systems is described in [135] where the technique tolerates one permanent fault.

FT Scheme 3: Re-execution

This FT scheme is based on the timing redundancy. A job is re-executed on the same processor when there is a fault during the first time computation. The job starts re-execution from the same initial state. This type of FT scheme incurs large time overhead which is not a suitable approach for hard RT applications with short deadline. The deadline of the application has to be maintained such that the RT properties are not violated. The scenario is shown in Figure 4.1 (c). The re-execution process is a recovery technique with single checkpointing.

FT Scheme 4: Checkpointing

Due to the large time overhead of re-execution process (as the job needs to run with complete CT to recover from a fault) a more complex FT scheme can be used to reduce the timing overhead. In this scheme the execution process is divided into several parts called as checkpointing. Different states of the execution of a job are stored or checkpoint and the job starts re-execution from that point when there is a fault. Thus checkpointing avoids a complete re-execution and takes less time to recover from a fault. The phenomena is shown in Figure 4.1 (d). Checkpointing can be made periodically either static or dynamic or it can be made depending on the states and knowledge of the application job. More checkpointing increases the complexity and checkpoint overhead on the other hand less checkpointing requires long execution time to recover (e.g., re-execution precess requires one checkpoint), thus a suitable number of checkpointing is necessary. An analysis of checkpointing for RT systems can be found in [97; 132]. The deadline of the application has to be maintained while performing checkpointing.

4.1.2 Influence and Communication Reduction

By allocating jobs with the highest mutual communication onto the same node, we strive to minimize the interactions and influences between jobs and the communication load on the physical network. The allocation of highly communicating jobs onto the same node is beneficial both for reducing the bandwidth and confining the inter job error propagation within a single node.

In doing this, it is important not to violate FT, timing and resource constraints. Communication clustering heuristics, which attempts to allocate highly communicating jobs to the same node, thus reducing the overall communication load on physical network, have been addressed in [84; 136]. As we endeavor to design an integrated embedded architecture where several nodes share a single network, the communication clustering heuristic is desirable. Between two communicating jobs, there is an influence that may lead to propagation of errors from one job to the other. When communication between two jobs is high, the influence between them is considered high as well. If a job is affected by an error of the node it is running on, it might propagate errors by interacting with jobs on other nodes. These interactions risk the failure of multiple nodes and may consequently lead to an overall system failure which are undesirable. Moreover messages sending over the network can cause loss of messages due to transmission error, e.g., in automotive cars EMI may cause communication failure due to transient errors. In Section 5.2.1 we describe a lot more on the influence including how to measure and estimate influence.

Example Describing the Benefits

We consider an example application (similar to [137]), which consists of four jobs j_1, j_2, j_3, and j_4. They need to be mapped onto an architecture consists of two nodes (n_0 and n_1) communicating via a network. The application and the architecture is shown in the upper part of Figure 4.2. All jobs must finish their execution by 140ms, i.e., by the deadline of the application. Individual CTs for each job are shown in the figure, e.g., job j_1 takes 40ms for the execution. Each job takes same amount of CT to execute on either processors. j_1 is a predecessor of j_2 and j_3, and sends messages m_{12} and m_{13} to j_2 and j_3 respectively. j_4 is a successor of j_2 and j_3, and receives messages m_{24} and m_{34} from j_2 and j_3 respectively. A TDMA based network is assumed for the communication where a TDMA round TD_x comprises two slots s_0 and s_1. For the purpose of deterministic message transmissions node n_0 and n_1 are statically assigned to slot s_0 and s_1 respectively. The slot length of the network is equal to 10ms and maximum 2 messages can be sent per slot. The time for intra-communication (communication within the same node) is assumed to be zero. This is shown in Figure 4.2 (b) when j_1 and j_3 are allocated on node n_0. These two jobs communicate through the services provided by the OS kernel layer while taking a negligible amount of time comparing with the time taken by the communication channel. We assume that the partition switching time is also negligible.

In the mapping configuration shown in Figure 4.2 (a), jobs j_1 and j_4

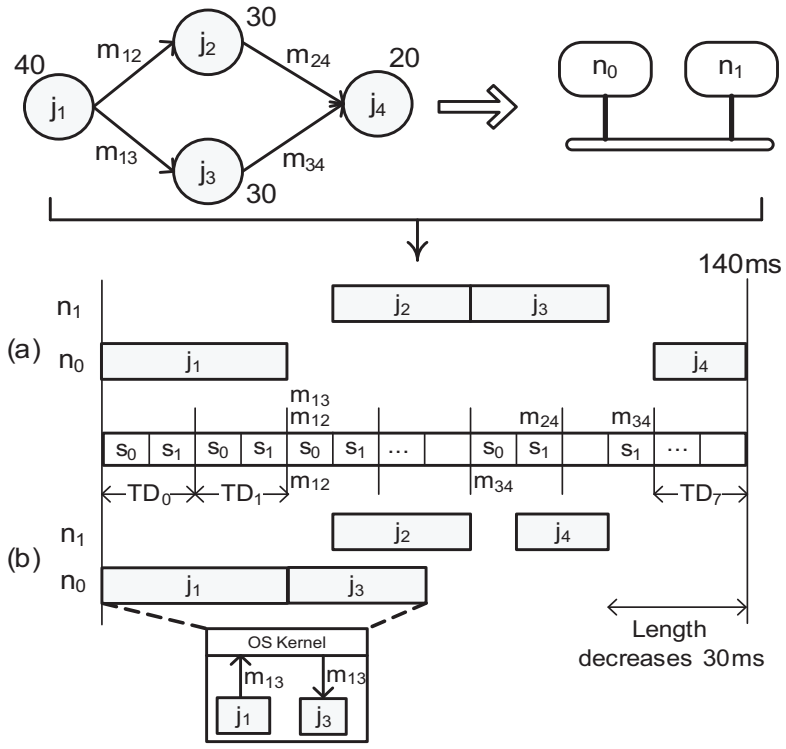

Figure 4.2: Reduction of influence and communication overhead

are assigned on node n_0 and j_2 and j_3 are assigned on n_1. As we see that this mapping takes 140ms to finish the computation of all the jobs which just satisfies the deadline and sends 4 messages over the network. This is due to directly interacting jobs are allocated on the separate nodes where job j_4 waits to start execution for the result of j_3. Whereas the mapping configuration shown in Figure 4.2 (b) (interacting jobs are on the same node) takes 110ms which is 30ms less than the previous mapping configuration and only 2 messages sends over the network. Therefore this configuration decreases the probability of messages loss over the network as well as reduces the error propagation probabilities between nodes.

We emphasize the following key benefits (where first two benefits enhance dependability) of assigning highly interacting jobs to the same node:

(1) Restricting the possible nodes from correlated faults,

(2) The probability of loosing messages over the network is reduced,

(3) The communication load on the network is reduced (may allow for the use of a slower but cheaper bus [67]), and

(4) Increases the over all performance by reducing the total execution time (computation time + time to send/receive messages) of a job since network delays are avoided.

Reducing Messages Loss – Perspectives

The transmission error, e.g., EMI may cause a severe threat to the correct behavior of the system functionality. In automotive systems, the EMI can either be radiated by in-vehicle electrical devices (switches, relays etc.) or come from a source outside the vehicle (radio, radars, flashes of lighting). EMI could affect the proper functioning of jobs by affecting the message transmission over the network and also could affect the correct functioning of any of the electronic devices, e.g., aboard a car. The transmission support is particularly weak link and the use of an all-optical network, which offers very high immune to EMI, is not generally feasible because of the low-cost requirement imposed by the industry. With a redundant transmission support such as in TTP/C, it is assumed that in case a single-bit or multiple-bits burst error is detected by cyclic redundancy checksum (CRC) on one channel, the redundant data on the unaffected channel is used. Even in this situation the network is not immune to transmission errors though provide some resilience to those errors [11]. A perturbation is likely to affect both channels in quite a similar manner because they are identical and very close one to each other. There is an ongoing research at NASA to avoid communication failure due to transient error caused by EMI. The proposed idea there is to use ROBUS (Reliable Optical BUS) [138] which is more immune to transient errors caused mainly by EMI. However the solution of using such network is very costly.

Other possibility could be on network protocol level such that either sending less number of messages over the network which reduces the probability of messages being corrupted or finding a way of using network such that less number of messages get corrupted even though there is a transient error on the communication channel. The second criteria is a suitable one for the replicas because replicas send messages over the network anyway as they are distributed over distinct nodes. When two replicas send messages in consecutive slots and if a transient error occurred between end of the first slot and beginning of the second slot then there is a probability of loosing messages from both of the replicas. Messages from two replicas can be sent such that a gap (preferably at least the length of the perturbation) is kept in the network between messages in order to minimize this probability of loosing messages from both replicas. If the message size is not equal to the length of the slot then first message can be sent at the beginning of the corresponding

node slot and the second message can be sent at the end of the assigned slot. Other possibility is to assign two replicas onto two different nodes where those nodes do not use consecutive slots.

4.1.3 Schedulability Analysis

Schedulability analysis deals with the timing analysis that provides the condition to satisfy the timing constraints of RT systems such that a valid scheduling can be found for a given set of jobs. The timing constraints (precedence and deadline) defined in Section 2.3.6 describe the RT properties of the system. In this section we provide a schedulability analysis to satisfy those constraints. In the mapping algorithm during the assignment of jobs the timing constraints are checked to assure that the allocation is schedulable. We assume that jobs may start executing from their earliest start time (EST) and takes a certain amount of CT which must finish before their deadline. If there are any precedence relations between jobs these must be preserved. When assigning jobs to a processor which already hosts one or more jobs, timing constraints are checked to ensure schedulability. The condition for schedulability presented in this thesis is necessary, and if preemptions are allowed, the condition is also sufficient. Since the proposed mapping takes place during the early stage of design, we do not enforce any particular scheduling strategy. Furthermore, there is no restriction on choosing either periodic or aperiodic jobs. As the underlying TDMA based time triggered communication base naturally supports periodic jobs, these are utilized in the examples. In general, most control applications specially SC applications such as the example in automotive area, often use periodic job sets.

For checking timing constraints, we let $\mathcal{J}$ be the set containing all jobs already assigned to that node, as well as the job we are about to assign. *If the difference between maximum end time (highest deadline) and minimum EST for any possible combination of jobs in $\mathcal{J}$ is less than the sum of their computation time, then we cannot assign the job to this processor.* If the resulting mapping is to be schedulable, the condition shown below (Equation 4.2) must hold for *all subsets of jobs* to be allocated on the same processor.

$$max_{\forall j \in \mathcal{J}}(D_j) - min_{\forall j \in \mathcal{J}}(EST_j) \geq \sum_{j \in \mathcal{J}} CT_j \tag{4.1}$$

Where $\mathcal{J}$ is the set of jobs that are being checked.

We now consider an example. Two jobs $j_3 \equiv \{EST, CT, D\} \equiv \{2, 4, 10\}$ and $j_4 \equiv \{7, 6, 14\}$ are already assigned to a given node. If we try to map

another job $j_5 \equiv \{5, 5, 10\}$ to this node, timing constraints are violated due to this job as the combination does not hold Equation 4.2. Scheduling of these jobs will not be possible (see Figure 4.3) since $((14 - 2 = 12) < (15 = 4+6+5))$. Thus j_5 must be assigned to a different node. Of course, the total computing time required by the jobs running on a single processor must not exceed the *computational capability* provided by that processor.

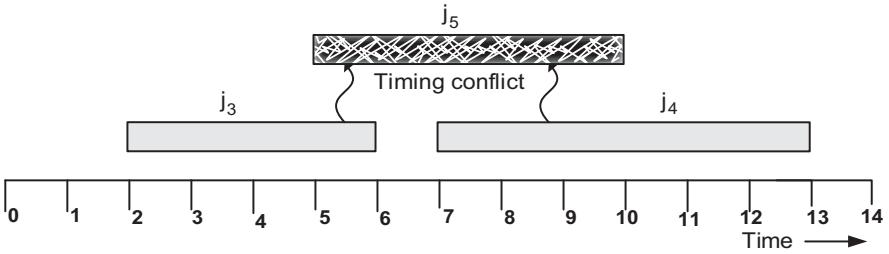

Figure 4.3: Schedulability analysis

For communicating jobs located on different processors the message transmission time through the network has to be considered to make the jobs schedulable. We assume a maximum network delay T_N for transmitting a message across the bus. This time must be bounded by using an appropriate protocol, e.g., a statically scheduled TDMA protocol [109]. Of course the network delay depends on whether a node gets access to a TDMA slot for sending messages in this TDMA round or will have to wait for the next round. The deadline of a job sending a message to another job located on a different node must be reduced by the time T_N to accommodate for possible network delays (see Figure 4.4). The message transmission time (T_N) estimation process is described later in this section.

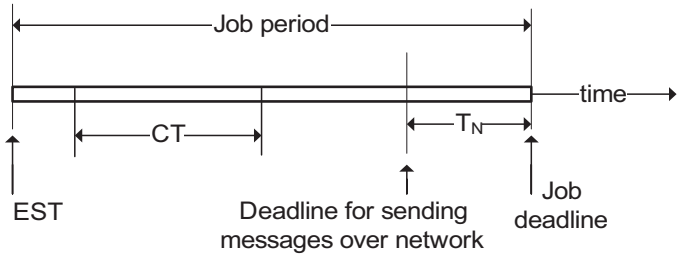

Figure 4.4: Network delay (T_N)

Scheduling for Integrated Approach

A hard RT system must execute a set of jobs in such a way that the time-critical jobs meet their specified deadlines [43]–page 228. Usually RT systems are divided into hard and soft RT systems, where scheduling of hard RT systems must be deterministic and have to satisfy hard deadlines. In hard RT systems often the jobs are periodic in nature (time triggered) and the scheduling for hard RT systems is thus made statically at design time. Whereas for scheduling of soft RT systems (event triggered) missing some deadlines may be acceptable for a degraded system performance. Moreover scheduling can be categorized as preemptive and non-preemptive. In a schedule if a running job is preempted, i.e., interrupted by another higher priority/important job then this is said to be a preemptive scheduling. On the other hand if a running job cannot be interrupted until it finishes the execution or releases the currently occupied resource, is said to be a non-preemptive schedule.

In traditional system design one function is assigned to a single node and therefore, typically, application jobs are scheduled on an independent processor, i.e., *uniprocessor scheduling*. Often the scheduling is also performed assuming an independent set of jobs. Such types of schedulability analysis and scheduling have been discussed in [139]. We now briefly describe some relevant scheduling techniques. One method for performing scheduling is the fixed-priority scheduling techniques where jobs are prioritized before scheduling, e.g., according to the shortest deadline or the earliest start time. Using this technique, [140; 141] present timing and schedulability analysis for hard RT periodic tasks model on a uniprocessor. Mostly the principal assumptions made were scheduling independent tasks/jobs onto a single processor. An efficient and exact method (sufficient and necessary condition of schedulability) is presented in [142] for the same type of fixed priority RT systems. However in our system design, jobs from different applications are assigned onto single processor and jobs from single application are assigned onto different processors, i.e., *a multiprocessor scheduling*. Moreover usually jobs have precedence relations among each other, i.e., the execution of one job depends on the result from the other. Consequently, new scheduling techniques are needed [26] which can consider distributed applications, data and control dependencies, and accurately take into account the communication protocols that have a significant influence on the timing properties. For such distributed RT systems, specifically the type of systems whose failure can be catastrophic due to violation of deadline, static/off-line scheduling algorithms are used to build, a schedulable table with activation and finish times for each jobs such that timing constraints are satisfied. An optimal task allocation and scheduling for communicating periodic hard RT tasks on a het-

erogeneous distributed platform is stated in [70]. A distributed scheduling for RT tasks on multi core platforms is described in [143], which is based on a cache-aware Pfair scheduling (specially used for periodic jobs scheduling). We have mentioned about various scheduling in the related work section. Particularly, we employ time-triggered scheduling which can be applied on a distributed platform/architecture consisting of multiple node processors. In order to validate the schedulability analysis we use an independent embedded time-triggered scheduling tool presented in [42].

TTP/TTX-Plan and TTP/TTX-Build (for TTP/C and TTX stands for FlexRay) are comprehensive commercial tools and part of the TTP/TTX-tools SW development suite [144; 145] employ scheduling techniques for distributed RT embedded systems. Using the time-triggered communication protocol (TDMA as communication scheme), TTP/TTX-Plan and TTP/TTX-Build tools derive the off-line schedule for messages and jobs respectively. Both of the tools exploit the advantages of time triggered architecture paradigm. These tools are the candidate set for our developed prototype to generate the scheduling.

Message Transmission Time

We provide an estimation of the message transmission time T_N through the network which has to be considered for communicating jobs located on different node processors to make the jobs schedulable. A similar message communication planning can be found in [109; 137] where the authors determine the slot and the specific TDMA round for a particular message to be sent. The estimation of T_N as well as the actual EST of allocated jobs are provided by the Algorithm 2 which is adopted from [109; 137]. The assumed time-triggered protocol provides deterministic access to the medium by ordering the message transmissions statically at design time and thus response time is guaranteed.

We use following parameters in the algorithm. n_k is the node which is ready to send the message over the network and s_k is a specific slot assigned to that node. b_{m_i} is the size of the message ready to send at time $release\_time$. Therefore, $release\_time$ is the message delivery time of a job. b_{s_k} is the size of corresponding slot where as $start_{s_k}$ is the starting time of the slot in a round for the corresponding node n_k. For a successor of more than one job, maximum delay caused by all its precedence jobs is used for calculating the actual EST and is computed as follows:

$$EST_i = max\left\{ready(j_i, n_k),\ max_{\forall j_j \in pred(j_i)}\left(ft(j_j, n_p) + T_{N_{i,j}}\right)\right\} \qquad (4.2)$$

Algorithm 2 Network delay calculation for messages transmission

1: **Function:** `message_transmission`$(n_k, b_{m_i}, release\_time)$
2: s_k `= assigned slot to` n_k`;` /*Slot assigned in TDMA round*/
3: $round = floor(release\_time/round\_length)$; /*Calculate current round*/
/*Next step checks whether the slot of the current round has passed*/
4: **if** $release\_time - round * round\_length > start_{s_k}$ **then**
5: $round = round + 1$; /*Increase a TDMA round*/
6: **end if**
7: **while** $b_{m_i} > b_{s_k} - b_{occupied}$ **do**
8: $round = round + 1$;
/*Increase round if the message size does not fit in this slot*/
9: **end while**
10: $T_{N_i} = round * round\_length + start_{s_k} + slot\_length - release\_time$;
/*Calculate the delay for a job depends on messages over the network*/
11: `return(`$round, s_k, T_{N_i}$`);`
12: `end message_transmission`

where $ready(j_i, n_k)$ is the earliest time at which processor n_k is ready to start executing the job j_i. $pred(j_i)$ is the set of all predecessor jobs of j_i. $ft(j_j, n_p)$ is the finish time of job j_j in node n_p computed by the sum of EST_j and CT_j. The message transmission delay between (j_i, n_k) and (j_j, n_p) is $T_{N_{i,j}}$.

After assigning the slot s_k to a respective node n_k in *Step 2* of the Algorithm 2, the TDMA round is determined where a node can possibly use its assigned slot. Whether a node can send messages using this slot is checked at *Step 4*. If the starting slot time $start_{s_k}$ of the corresponding TDMA round has passed comparing to the message release time ($release\_time$) then the next TDMA round is selected for the message to be sent. If the size of the messages is bigger than the size of the corresponding slot (which is checked in *Step 7*) then some messages are sent in the next TDMA round. The overall delay due to the message transmissions over the network is calculated for each successor job at *Step 10*.

Estimation of T_N and EST: For inter-job communication, it is necessary to calculate the network delay. This delay depends on the type and speed of the network. The function shown in Algorithm 2 is used for calculating T_N and actual EST. In order to illustrate the determination process of actual EST we use the same example and mapping configurations depicted in Figure 4.2. We describe estimation of ESTs for the jobs which are having precedence relations. We choose the mapping configuration of Figure 4.2 (b) where more messages are sent over the network. Node n_0 and n_1 are assigned to slot s_0 and s_1 where the slot starting time $start_{s_0}$ and $start_{s_1}$ is 0ms

and 10ms respectively for every TDMA round where TD_x = 20ms. We progressively calculate T_N and EST for each predecessor job. We calculate the actual EST for job j_2 which is dependent on j_1 and can only start execution once it gets message from j_1. These two jobs are mapped onto different nodes. T_N needs to be calculated by the delay caused by the message received from j_1 which is assigned onto node n_0. The $release\_time$ of j_1 is 40ms as it starts execution from 0ms. In this case the TDMA round will be 2 and the condition in *Step 4* is not satisfied thereof no need to increase the TDMA round. Thus the node will send the messages m_{12} and m_{13} of j_1 from this slot as the slot is unused yet and has the capacity of sending two messages at the same time. j_2 has to wait 10ms which is the T_N (calculated at *Step 10*). If the size of the slot would have been smaller than the two messages size then it has to wait another round to send the next message. The actual EST of j_2 will be $(40 + 10) = 50$ms (calculated from the Equation 4.2). In this way we can estimate the actual EST of j_4 considering all the previously assigned jobs which is 120ms.

4.1.4 Resource Consumption

We explain the memory resource consumption denoted by M_r. Each job has its own memory requirement. The total amount of memory required by all jobs which are allocated to the same processor, has to be supported by the memory capacity available for jobs on that processor. The utilization of memory available for jobs on a processor should be less than or equal to 100%. The following equation describes this relationship, where m_i = memory used by the i^{th} job allocated to the k^{th} node processor, mc_k = memory capacity of that processor and n is the number of total jobs:

$$M_r \;=\; \forall_{i,k} \frac{\sum_{1<i\leq n} m_i * M_{i,k}}{mc_k} \leq 1 \tag{4.3}$$

where,

$$M_{i,k} \;=\; \begin{cases} 1 & \text{if } i^{th} \text{ job is assigned to } k^{th} \text{ node,} \\ 0 & \text{otherwise.} \end{cases}$$

However the above memory strategies must not necessarily be hold in every cases, e.g., if two or more processors share a memory device then the memory capacity per processor must not be fixed *a priori*.

4.2 Supporting Data Structure

We introduce matrices for the purpose of ease structuring and implementing the ordering heuristics and the mapping algorithm. The *allocation compatibility matrix A* is used to check the usable nodes for each job and accordingly jobs and nodes are ordered (Job ordering *1a* and *1b* – Section 4.3.1). The *communication matrix C* represents the communication between jobs and is used to determine the most communicating jobs (Job ordering *2a* and *2b*). These matrices are discussed in the following.

Allocation Compatibility Matrix A

A rectangular matrix $A_{k\times n}$ is used to describe the possible assignment of a single job onto nodes, in such a way that rows represent nodes and columns represent jobs, where k is the total number of nodes and n is the total number of jobs. Note that all replicas of the same job are represented using only one column. Each element of the matrix is filled with either 0 or 1, 1 if a job j_i can be assigned to a node n_k and 0 if it cannot. Restrictions on which nodes a job can be assigned to is the result of binding constraints and are mostly determined by the use of particular resources, e.g., when a job needs sensors or actuators.

Some special criteria are listed below (but not restricted to) for deciding whether a job can be assigned to a node or not due to its binding functionality.

- When a job needs sensors or actuators,
- If a job needs any special I/O devices and peripherals,
- If DSP, ASIC or HW watchdog are needed by jobs, etc.

Communication Matrix C

A communication matrix of size $n \times n$ is used in order to determine the most communicating jobs. Each element of the matrix corresponds to the communication of a pair of jobs, and n being the number of jobs (counting replicas of the same job only once). If there is communication between two jobs i and j, we use the value $C_{i,j} \equiv (b_{i,j})$ to represent the total amount of data (bytes) being transferred. If there is no communication, 0 is used. This means that the communication matrix by construction will be square symmetric (the values of main diagonal are zero and the values of upper-triangular and lower triangular are similar, see Table 4.2). Note that $C_{i,j}$ denotes the maximum amount of communication possible between jobs i

and j for one time execution (i.e., the available size of sent and received messages as defined by the system user) in a period.

Mapping Matrix M: The mapping matrix is an $n \times m$ binary matrix, which represents the mapping of n jobs onto m nodes. Each element $M_{i,k}$ of the matrix is filled with either 1 or 0, 1 if a job j_i has been allocated to the node n_k and 0 otherwise. This matrix has been implemented to represent the resulting mapping and have used to check whether a job already has been assigned to a node or not.

4.3 Ordering Heuristics

Ordering heuristics are used to create feasible mapping for a reduced number of *backtracking* or even no backtracking is necessary if an optimal ordering can be obtained [50]. The idea behind our ordering heuristics is to judiciously determine the ordering of jobs and nodes in order to facilitate the recursive assignment. Jobs are ordered so that the most conflicting and most constrained jobs are handled first. Similarly, the nodes which allow the most assignments are ordered first. During the mapping algorithm, we start by assigning the first job from the ordered list onto the first node from the ordered nodes and continue until all jobs have been assigned.

The assignment process needs two important heuristics for having better performance of solving the mapping problem, namely how to decide which job to assign next (ordering of jobs), and which node to assign to this job (ordering of nodes). This is similar to so called variable[1] (job) and value (node) ordering heuristics which are concerned with the order in which variables are instantiated and values are assigned to each variable. A good variable ordering is one that starts with the variables that are the most difficult to instantiate (i.e., most constraining variable ordering heuristic) and a good value ordering heuristic is one that leaves open as many options as possible to the remaining uninstantiated variables (i.e., a so-called least constraining value ordering heuristic). Using an example we provide more reasonings on the ordering heuristics. These ordering heuristics can have a great impact on search efficiency. No backtracking would be necessary if an optimal variable/value ordering is achieved [50], thus in such a case a linear time solution for the mapping problem is possible. Also, [146] shows that a proper and good selection of ordering can reduce, on average, the number of steps/iterations to find a feasible solution. Consequently, a bad selection or no selection of ordering may increase the number of iterations as well as the search space.

[1]The meaning of this variable is not the same as used in the optimization and other part of the thesis. This meaning [50] is only used in this section.

There are several heuristics one can consider for the ordering, e.g., consider those jobs first which participate in the highest number of constraints [146], ordering jobs according to their combined resource consumption (computation, memory and bandwidth) in descending manner [37], or ordering them according to their importance, or importance of their properties [79].

Therefore, a proper and good selection of ordering is essential and can reduce the number of iterations to find a feasible solution. For creating the mapping, we propose job and node ordering heuristics which are described in the subsequent sections. The overall aim of the ordering heuristics is: *Generation of a list of the most conflicting (jobs that cannot be mapped on the same node, e.g., replicas) and the most important jobs (if a job needs any binding functionality) which can easily guarantee the feasibility of the generated initial solutions.*

Justification of Ordering Heuristics

The algorithm presented in Section 4.4 considers the most important and conflicting jobs first for the assignment which can ease the creation of feasible mappings. This consideration is realized by explaining the following example.

We consider four jobs j_1, j_2, j_3 and j_4 and their allocation onto nodes n_1 and n_2. Job j_2 is a high critical job and is replicated twice j_{2a}, j_{2b} according to its degree of criticality. We assume that two jobs (not the replicas) can run on one node and three jobs can run on another node with sufficient resources, i.e., a node can host maximum three jobs. Four types of constraints $\{c_1, c_2, c_3, c_4\}$ are defined which need to be satisfied during the assignment process: c_1 – due to the binding functionality j_1 must run on n_1, c_2 – j_{2a} and j_{2b} must run on separate nodes in order to tolerate faults, c_3 – due to computational capability and c_4 – due to memory resource capacity. The type c_3 and c_4 are common/general constraints that are always required by each job and present in each allocation problem. Since they refer to all jobs, do not give us a direct hint which jobs should be assigned first. On the other hand, constraint c_2 explicitly refers to conflicting jobs, i.e., replicas j_{2a} and j_{2b} cannot be assigned onto the same node and c_1 exclusively mentions about importance of j_1 which requires a sensor. Thus, j_1 and j_2 are to be considered first for the mapping. Therefore we start by assigning j_1, j_2 and then j_3, j_4 as follows: *(a)* assign j_1 onto n_1 as enforced by c_1, *(b)* j_{2a} and j_{2b} are assigned to n_1 and n_2 due to constraint c_2, *(c)* j_3 can now be assigned to either node, we arbitrarily choose n_1 and *(d)* j_4 must be assigned to n_2 (due to the resource constraints of the nodes). Now, assume an arbitrary job ordering of $j_1, j_3, j_4, j_2(j_{2a}, j_{2b})$. The assignments are as follows: *(a)* assign j_1 onto n_1 as enforced by c_1, *(b)* j_3 can be assigned to either node, we arbitrarily

choose first node n_1, *(c)* j_4 can be assigned to any node, we choose n_1 again, *(d)* as enforced by c_2, j_{2a} and j_{2b} have to assign on different nodes but we fail to assign on n_1 due to c_3 and c_4, therefore, *(e)* backtracking (or back jumping to step *(b)* and *(c)*) is necessary so that either j_3 or j_4 can be moved to n_2 to create a feasible assignment. When job j_1 requires binding functionality is considered last in the order list backtracking is also necessary. Of course the proposed ordering is no guarantee that backtracking will never be necessary. However we believe that the proposed *a priori* heuristics for ordering jobs and nodes is a viable strategy for developing the SW-HW mapping.

4.3.1 Job Ordering

Job ordering heuristics are used to order the jobs in a list, i.e., to decide which jobs to assign first. The compatibility matrix A and the communication matrix C presented in the previous section is employed in the ordering as follows:

1a. Create a sub-matrix $\widetilde{A}$ of the assignment compatibility matrix A, containing only those jobs (columns) to be assigned in a *Phase* of the allocation[2]. If a job can be assigned to a node the corresponding cell of $\widetilde{A}$ is set to 1 otherwise 0.

b. Sum each column (representing a job) in the matrix $\widetilde{A}$. Order the jobs in ascending order, i.e., the jobs with the least possible assignments will come first. By considering these jobs first, the search space is likely to decrease since these jobs are the most constrained (with respect to binding constraints). Ties are broken according to the second heuristic given below.

2a. Create a sub-matrix $\widetilde{C}$ of the communication matrix C, containing only those rows and columns belonging to jobs that are to be assigned in this specific Phase. For *Phase II*, a sub-matrix $\widetilde{C}$ of the communication matrix C is created, containing both the rows and columns belonging to jobs that are to be assigned in this *Phase*, as well as those rows and columns belonging to jobs assigned in *Phase I*. The reason for including already assigned jobs in the matrix $\widetilde{C}$ in *Phase II*, is that jobs to be assigned in this phase belong to SC applications and thus are more likely have communication with the jobs previously assigned in *Phase I*.

[2] *Phases I, II and III* are used in the mapping Algorithm 3 described in Section 4.4.

b. Search the matrix $\widetilde{C}$ and find the pair of jobs with the highest mutual communication between them. Arbitrarily, select one of the jobs in the pair and order that job first, followed by the second job in the pair. If any (or both) of the jobs in the pair have already been ordered, just ignore it. Continue with selecting the next most communicating pair and order those jobs as described, until there are no jobs left. Ties are broken arbitrarily. This heuristic can be applied stand-alone when jobs are not restricted by binding constraints.

Note that for implementing these heuristics, it is not necessary to create the full compatibility matrix A nor the full communication matrix C, the sub-matrices ($\widetilde{A}$ and $\widetilde{C}$) will suffice. Further the sub-matrix of the communication matrix, which is used in *Phase I* is itself a part of the sub-matrix used in *Phase II*. Hence, the sub-matrix of *Phase II* could be created and used in *Phase I*, reducing the number of matrices that need to be created. Also, the symmetry of the communication matrix (and its sub-matrices) can possibly be exploited in the implementation. In this case the search for the highest mutual communication pair relies either only upper or lower triangular part of the matrix $\widetilde{C}$.

4.3.2 Node Ordering

Just as in the job ordering heuristics, the same sub-matrix $\widetilde{A}$ of the allocation compatibility matrix A is used for ordering nodes. Nodes are ordered by taking the sum of each row (representing nodes) in the sub-matrix $\widetilde{A}$, and ordering the nodes in descending order. By using this ordering the nodes which allow the most assignments are ordered first. Ties are broken arbitrarily.

Example Describing the Ordering Heuristics

We take an example which consists of four jobs j_1, j_2, j_3, j_4 and two nodes n_0, n_1. Job j_2 needs a sensor and node n_1 has a sensor attached to it. Let us assume that the mutual communication volume within a time period between j_1 and j_2 is 4 bytes, between j_1 and j_3 5 bytes, between j_2 and j_4 8 bytes and between j_3 and j_4 is 5 bytes. When the assignment compatibility matrix A is created (Table 4.1), we see that job j_2 can only be assigned to node n_1, correspondingly n_1 is the only usable node for j_2. Since j_2 cannot be assigned onto n_0, we put 0 in the corresponding cell. All other jobs can be assigned to any of the two nodes. We put 1 in the corresponding cell when a job can be assigned onto a node. The ordering of jobs will start by j_2 and node n_1 will come first in the node ordering. Ties are broken for other jobs by using the

A	j_1	j_2	j_3	j_4	$\sum$
n_0	1	0	1	1	**3**
n_1	1	1	1	1	**4**
$\sum$	**2**	**1**	**2**	**2**	

Table 4.1: Building assignment compatibility matrix

C	j_1	j_2	j_3	j_4
j_1	**0**	4	5	0
j_2	4	**0**	0	8
j_3	5	0	**0**	5
j_4	0	8	5	**0**

Table 4.2: Building communication matrix

communication matrix (Table 4.2). Each cell of the communication matrix is filled by the total amount of mutual communication volume in bytes of the corresponding jobs pair. The upper and lower triangular parts of this matrix is symmetric. Hence during the implementation we only need to search for the communication pairs either in upper or lower triangular part. We see that job j_2 and j_4 have high mutual communication among all the pairs followed by the pair j_4, j_3 and j_3, j_1. Hence the job ordering will be j_2, j_4, j_3, j_1 and the node ordering will be n_1, n_0. If a job appears more than once in different pairs then from the first pair it is placed in the ordered list. If we assume that maximum two jobs can be assigned on a single node due to the computation and resource constraints then node n_1 will host the jobs j_2 and j_4 and n_0 will host the jobs j_3 and j_1. The replicas are not included in either matrices only the primary job is included. When a job is replicated two times the corresponding communication link of the job is also replicated and the communication volume becomes double. Only the primary replicas (the main job) are considered in the communication matrix for their ordering. The other replicas are not considered as they will be assigned onto different nodes anyway and will disseminate messages over the network.

4.4 The Mapping Algorithm

On the background of particularly the SW and HW models, the constraints set and on the premises of mapping policies (separation of replicas, schedulability, reduction of communication and influences) and ordering heuristics, we now present the mapping algorithm. The goal of the mapping is to assign jobs onto available HW resources satisfying all the defined constraints. The construction of the algorithm is inspired by the established *constructive heuristics* in space allocation [48], in course timetabling [49] and by the ordering heuristics of the job shop scheduling constraint satisfaction problem [50]. The mapping algorithm employs the constraints handling techniques as well

as the search techniques presented in Section 3.3. The algorithm works in

Algorithm 3 Extra-functionality driven SW-HW mapping algorithm

Input: Set of jobs and nodes
Output: Mapped jobs onto nodes

1: Let J be the set of all jobs to be assigned in this phase.
2: Replicate jobs according to their degree of criticality, so that J now contains a set of replicas. Replicas are represented as $j_{ia...z}$ (e.g., if job j_1 is replicated two times then it is represented as j_{1a}, j_{1b}).
/*Replication only occurs in *Phase I*/
3: Order the jobs according to a job ordering heuristics, and let J represent the list of ordered jobs.
/*Section 4.3.1 provides a discussion of the heuristics used*/
4: Order the nodes according to a node ordering heuristics, and let N represent the list of ordered nodes.
/*Section 4.3.2 provides a discussion of the heuristic used*/
5: Select the first job $j_i \in J$.
6: Select a node $n_k \in N$ that has not been evaluated already as a possible assignment for j_i.
7: Evaluate if the selected job j_i can be assigned to node $n_k \in N$. If the assignment is possible, then assign j_i to node n_k, else go back to *Step 6*.
/*See Section 4.4.1 for a detailed discussion*/
8: If the assignment was successful proceed to *Step 9*, else a dead-end has been reached (i.e., a valid assignment cannot be found for j_i). When a dead-end is reached then backtrack, i.e., undo one or more previous assignments and try alternative ones. If no feasible solution is found by backtracking, then report that the mapping is infeasible and terminate.
/*The backtracking goes back to the next most recently instantiated job, i.e, chronological backtracking*/
9: Remove the allocated job j_i from the list J and then repeat (from *Step 5*) the same procedure until the list J is empty.

three phases and considers SC applications and non-SC applications separately to reduce influences. As a result of component based design, SC and non-SC applications communicate minimally, thus they can be treated separately. To facilitate strong partitioning between SC and non-SC applications, we allow that jobs of SC applications and jobs of non-SC applications can be allocated onto separate processors or processor cores on the same node. The jobs are assigned in three different phases, mentioned below:

Phase I: High critical jobs of SC applications,

Phase II: Non-replicated jobs (if any) of SC applications, and

Phase III: Jobs from non-SC applications.

The prescribed algorithm is executed once in each *Phase* of the mapping process. We start by considering the most conflicting jobs that cannot be mapped on the same node (i.e, replicas) in the first phase. Throughout the process the most constrained jobs (with respect to binding constraints B_f) are assigned first. Using this ordering and assigning replicas in *Phase I*, the number of backtracks are reduced (see experimental results in Section 6.2). In *Phase II*, we continue with non-replicated jobs of SC applications, they will be integrated with the replicated jobs of SC applications in a way that reduces job influences. As the lower critical jobs from SC applications are treated in a different phase, it is more likely that there will be less influences between them. Finally, jobs from non-SC applications are allocated in the third phase. A high level description of each mapping phase is outlined in Algorithm 3.

4.4.1 Assignment Evaluation

An evaluation has to be performed *a priori* a job can be assigned to a node. Different techniques are applied to satisfy the constraints while assigning jobs onto nodes and to explore the search space. In this step of the mapping, all the defined constraints h_C are satisfied according to the procedure shown in Algorithm 4. If the node is empty, i.e., there are no previously assigned jobs to that node, then only binding constraints need to be checked. If there are already assigned jobs on a node then apply *retrospective techniques*. If all constraints (B_f, fault tolerance F_t, schedulability S_t and memory resource consumption M_r) hold, i.e., consistency enforcing is ensured then the next job is selected. While assigning jobs during *Phase I*, different nodes are selected for the replicas in *Step 6*, which significantly reduces the number of iterations as well as the number of backtracks to find the feasible solution. If the verification of consistency fails, exploration continues with the next node. When all nodes for this job has been checked unsuccessfully, the backtracking goes back to the most recently instantiated job, and so on (*Step 8* of Algorithm 3). Backtracking is simply performed by swapping or moving the jobs between nodes. After performing a move if a feasible assignment is found then the algorithm is continued with selecting the next job from the actual list. If a solution is not found after sufficient backtracking then the algorithm returns an infeasible mapping.

4.4.2 Constraints Satisfaction Technique

Algorithm 4 Satisfaction of constraints during the mapping

```
1: Function: Constraints h_C satisfier C(h_C)
2: Initialize bool: B_f, F_t, S_t, M_r, h_C, Feasibility;
3: h_C = {B_f, F_t, S_t, M_r};
4: Let i^th job is instantiated to assign onto k^th node;
5: i also associates the corresponding job ID at this point;
6: repeat
7:    B_f = bindFunctionality(i, k);
      /*Checking the binding functionality B_f ∈ h_C*/
8:    if (B_f == true) then
9:       F_t = faultTolerance(i);
         /*Assuring replicas are not in the same node, F_t ∈ h_C*/
10:      if (F_t == true) then
11:         S_t = schedulability(i);
            /*Checking timing constraints, S_t ∈ h_C*/
12:         if (S_t == true) then
13:            M_r = memoryConsumption(i);
               /*Memory constraints checking, M_r ∈ h_C*/
14:         end if
15:      end if
16:   end if
17:   if ((B_f && F_t && S_t && M_r) == true) then
18:      return h_C = true;
19:   else
20:      h_C = false;
21:   end if
22:   if ((h_C == false)&&
      (all the nodes have been explored for i^th job)) then
23:      backtrack = changeAssignment(i);
         /*Changing the allocation*/
24:      if ((B_f && F_t && S_t && M_r) == true) then
25:         return h_C = true;
26:      else
27:         Feasibility = false;
28:         Failed to satisfy constraints, i.e., h_C = false
29:         return Infeasible mapping;
30:      end if
31:   end if
32:   if ((the list J is empty) && (h_C == true)) then
33:      Feasibility = true;
34:      return Feasible mapping;
35:   end if
36: until ((!B_f) || (!F_t) || (!S_t) || (!M_r))
```

The constraints satisfaction algorithm (Algorithm 4) is implemented at

Step 7 of the Algorithm 3. It is also applied while performing the backtrack whenever necessary until all the constraints h_C are satisfied. If any one of the constraints is not satisfied then the next $(k+1)^{th}$ node is selected for assigning the current i^{th} job. All the nodes are visited in this way. If the constraint is still not satisfied then backtracking is necessary. The condition whether all the nodes have been explored by a selected job is controlled by the Algorithm 3.

4.4.3 Remarks

In the case when non-SC applications share the same processor as SC applications, some optional strategies are possible. After the jobs belonging to SC applications have been assigned, nodes can be re-ordered in a way that eases the assignment of non-SC jobs. As an example, ordering the jobs according to the amount of remaining computation capacity of each node thus better load balancing between nodes can be achieved. Another possibility is to re-order the nodes according to least memory utilization or nodes having less failure rate (useful when heterogeneous platform is assumed). All of these re-orderings might also be beneficial from a dependability viewpoint. Since non-SC jobs will be assigned primarily to nodes with few/no jobs from SC applications, the separation of SC and non-SC jobs is likely to increase. This will reduce the likelihood of errors occurring in non-SC applications propagating to SC applications.

Since each node can host multiple jobs from different applications, we can consider that the jobs should be assigned according to the physical placement of HW nodes in order to reduce the wiring and communication overhead. For example, the node located in the right front of the car can be allocated by the jobs of right front brake-by wire system and by the jobs of right front window lifter of doors subsystems.

4.5 Mapping Illustration

In order to illustrate the mapping process, we present real-life example of automotive applications, consisting of one SC and one non-SC application. We consider a HW architecture where each node (HW-C) comprises dual core processors for the computation. To the best of our knowledge, no mapping processes consider architectures making use of separate processor cores for SC and non-SC applications. However this type of architecture might be required for implementing highly dependable systems. To show the uniqueness of our approach, in this illustration, we consider such an architecture. The

scheduling of such architecture can be found in [143]. We start by describing the HW architecture onto which the jobs are mapped. Next, we describe the applications forming our example and the jobs they consist of. We continue by showing how the jobs of the applications are mapped onto nodes in the architecture during different *Phases* of the mapping algorithm. Finally, we summarize the results of the mapping.

4.5.1 HW Resources and Applications

The available hardware architecture for this illustration consists of four nodes (n_1, n_2, n_3, n_4) each containing two processor cores. During the assignment process we take the decision of assigning SC applications in one core and non-SC applications on another core. The nodes are connected to a physical network. All nodes have identical processors with equal computing power and the same amount of memory and bandwidth. The memory capacity of each processor of a node is $15MB$ and nodes are connected to a communication channel with a bandwidth of $10kb/s$. Node n_3 has a temperature sensor attached to it.

A brake-by-wire [147] system is used to represent a typical SC application. The non-SC application is a door control system extracted from an actual FIAT vehicle specification. Values for the different jobs (SW-Cs) properties of the applications have been chosen to illustrate the mapping algorithm.

Safety Critical Application

The brake-by-wire system in a car is a SC application that must provide services despite of any perturbations. The considered brake-by-wire system consists of six jobs $j_1, j_2, \ldots, j_6$, namely the brake pedal sensor (BPS) which produces the pedal signal. The brake force control (BFC) uses the pedal signal to calculate the brake forces for the four actuators (BAFR-brake actuator front right, BAFL-brake actuator front left, BARR-brake actuator rear right, BARL-brake actuator rear left). BPS j_1 and BFC j_2 are SC jobs and are replicated three times to run in a TMR setting. A structural model of the brake-by-wire application is shown in Figure 4.5 (a) (before and after replication). For simplicity, only one brake actuator (BA) is shown in the figure. The values of job properties are shown in Table 4.3. In the communication column the entry $50 \rightarrow j_2$ indicates that the corresponding row job j_1 sends 50 bytes of data to job j_2.

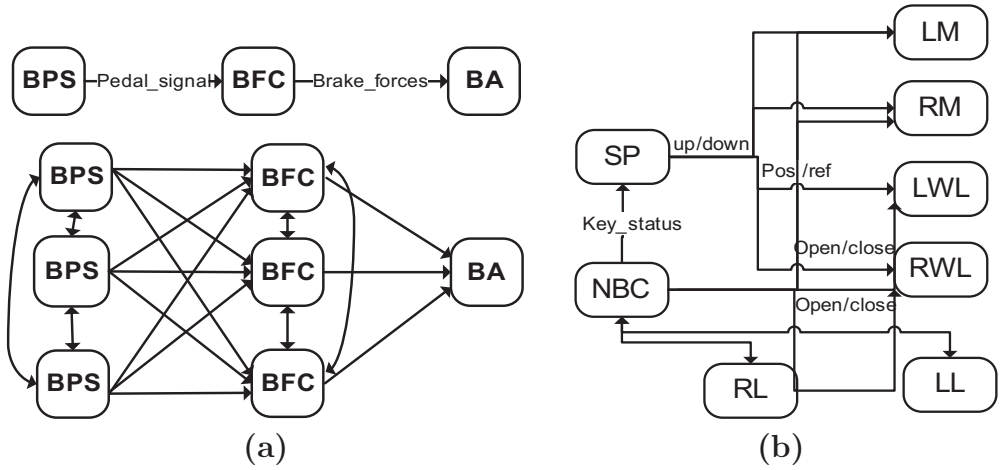

Figure 4.5: Brake-by-wire application (a) and doors application (b).

Job list	Name	EST (ms)	CT (ms)	D (ms)	Memory (MB)	Communication (bytes)
j_{1a}	BPS	0	5	14	5	$50 \rightarrow j_2$
j_{1b}	BPS	0	5	14	5	$50 \rightarrow j_2$
j_{1c}	BPS	0	5	14	5	$50 \rightarrow j_2$
j_{2a}	BFC	2	4	12	3	$60 \rightarrow j_3, 60 \rightarrow j_4$ $80 \rightarrow j_5, 80 \rightarrow j_6$
j_{2b}	BFC	2	4	12	3	$60 \rightarrow j_3, 60 \rightarrow j_4$ $80 \rightarrow j_5, 80 \rightarrow j_6$
j_{2c}	BFC	2	4	12	3	$60 \rightarrow j_3, 60 \rightarrow j_4$ $80 \rightarrow j_5, 80 \rightarrow j_6$
j_3	BAFR	4	7	17	4	
j_4	BAFL	4	6	14	4	
j_5	BARR	7	6	20	6	
j_6	BARL	5	7	18	6	

Table 4.3: Chosen values of job properties (brake-by-wire system)

Non-Safety Critical Application

The doors control application (assuming a two doors car) is a non-SC application which controls the closing of the doors and windows as well as the heating of the mirrors. We decompose this application into following 8 jobs $j_7, j_8 \ldots, j_{14}$. A switch panel (SP) detects whether the switches are pressed or released and sends the corresponding commands to the window lifter and mirror. The mirror handler consists of two jobs - one for the left mirror

and one for the right mirror (LM and RM) which are designed for moving, heating and comfort closing of mirrors. These two jobs require temperature sensors (TS), i.e., they have binding functionality (B_f). The window lifter consists of two jobs for lowering and raising the windows, left front window lifter (LWL) and right front window lifter (RWL). The lock-unlock function is responsible for locking, unlocking and dead-lock functionality of the door. It comprises two jobs: right lock-unlock (RL) and left lock-unlock (LL). The body computer node (NBC) coordinates the operation of the other jobs (mirror, window lifter and lock-unlock) and sends the status information to the switch panel (SP). A structural model of the doors application is shown in Figure 4.5 (b). The values of job properties are outlined in Table 4.4.

Job list	Name	B_f	EST (ms)	CT (ms)	D (ms)	Memory (MB)	Communication (bytes)
j_7	NBC		0	7	15	7	$40 \rightarrow j_8$ $30 \rightarrow j_{13}, j_{14}$ $30 \rightarrow j_9, j_{10}, j_{11}, j_{12}$
j_8	SP		0	6	16	8	$40 \rightarrow j_9, j_{10}, j_{11}, j_{12}$
j_9	LM	TS	4	4	13	3	
j_{10}	RM	TS	4	4	12	3	
j_{11}	LWL		3	8	17	6	
j_{12}	RWL		3	8	17	6	
j_{13}	RL		5	5	20	4	$30 \rightarrow j_7$
j_{14}	LL		5	5	20	4	$30 \rightarrow j_7$

Table 4.4: Chosen values of job properties (doors application)

4.5.2 Illustration of Mapping Phases

We proceed by describing the allocation of jobs that takes place during the three phases of our mapping algorithm depicted in Section 4.4. We demonstrate the assignment of jobs from above stated two applications onto the nodes of considered architecture.

Phase 1: Assignment of Replicated Jobs of SC Applications

In *Phase 1*, we consider the high-criticality jobs from SC applications. In this example the high-criticality jobs are j_1 and j_2, which both have a degree of criticality equal to three. We replicate these jobs to get all the jobs that need to be assigned in this *Phase*. This gives us the following jobs to consider:

$j_{1a}, j_{1b}, j_{1c}, j_{2a}, j_{2b}, j_{2c}$. We then proceed by creating the allocation sub-matrix $\widetilde{A}$ and the communication sub-matrix $\widetilde{C}$, which may assist in the ordering of jobs and nodes. As none of the jobs require any special features, they can both be assigned to any node. Hence, the sub-matrix $\widetilde{A}$ relevant for *Phase 1* will contain only ones. Using the first job ordering heuristic results in a tie. Thus, the jobs are ordered using the second heuristic. Since, we are only assigning two jobs (replicas are not considered in the communication matrix). Using the communication heuristics also results in a tie. The jobs are thus ordered arbitrarily, lets assume they are ordered as $j_{1a}, j_{1b}, j_{1c}, j_{2a}, j_{2b}, j_{2c}$.

Now, we try to order the nodes using the sub-matrix $\widetilde{A}$. The result is a tie, since all jobs can be assigned to all nodes. This means that the nodes can be ordered arbitrarily, lets say that they are ordered as n_1, n_2, n_3, n_4. We then select job j_{1a} and successfully try to assign it to node n_1. Then we select job j_{1b} and try to assign this to node n_1 as well. However the node assignment evaluation (*Step 7* of the Algorithm 3) shows that this is impossible since it violates the requirement that replicas should be separated as a requirement for FT. Consequently, job j_{1b} will be assigned to the next node n_2 and job j_{1c} will be assigned to node n_3. Similarly, the three replicated jobs of j_2 will be assigned to the first three nodes. However in order to reduce the search space for large number of jobs we take different nodes for replicas to be assigned. The node assignment evaluations show that these assignments can be performed without any violation of constraints. This concludes *Phase 1*.

Phase 2: Assignment of non Replicated Jobs of SC Applications

In *Phase 2*, we try to assign the non-replicated jobs of SC applications. In the example, this means that jobs j_3, j_4, j_5 and j_6 are needed to be assigned. Again, the assignment compatibility sub-matrix $\widetilde{A}$ cannot be used for deciding the job ordering. Thus we use a sub-matrix $\widetilde{C}$ of the communication matrix C, to decide the job ordering. This reduces interactions and communication load on the network. The used matrix is shown in Table 4.5. As discussed in the job ordering heuristics section, this sub-matrix also contains the already assigned jobs j_1 and j_2. Applying the communication heuristics we come up with the following job ordering: j_5, j_6, j_3, j_4 (with ties broken according to lowest index first).

The nodes are already ordered. Job j_5 can be assigned to node n_1 without violating any constraints. The schedulability analysis shows that job j_6 cannot be assigned to n_1 due to a timing constraint violation. The highest/maximum deadline among those jobs assigned (j_{1a}, j_{2a}, j_5) and the job (j_6) about to be assigned to n_1 is 20 and the lowest earliest start time is 0. Thus, the difference is 20, while the sum of computation times is

Job list	j_1	j_2	j_3	j_4	j_5	j_6
j_1	**0**	50	0	0	0	0
j_2	50	**0**	60	60	80	80
j_3	0	60	**0**	0	0	0
j_4	0	60	0	**0**	0	0
j_5	0	80	0	0	**0**	0
j_6	0	80	0	0	0	**0**

Table 4.5: The sub-matrix $\widetilde{C}$ used in *Phase 2*

$(5 + 4 + 6 + 7) = 22$, which is greater than 20. In this case we applied the schedulability analysis presented in Section 4.1.3. Consequently, job j_6 is assigned to the next explored node, i.e., to n_2. Since there is not enough memory capacity, job j_3 cannot be assigned to any of the nodes n_1 and n_2. Also, assigning j_3 to any of the nodes containing replicas of j_1 and j_2 would violate timing constraints. Therefore, job j_3 will be assigned to node n_4. The same reasoning holds for j_4, which is also assigned to the node n_4. The allocation resulting from the execution of *Phase 1* and *Phase 2* for the SC application is shown in Table 4.6.

n_1	n_2	n_3	n_4
j_{1a}, j_{2a}, j_5	j_{1b}, j_{2b}, j_6	j_{1c}, j_{2c}	j_3, j_4

Table 4.6: Jobs allocation of the brake-by-wire application

Phase 3: Assignment of Jobs of non-SC Applications

In this *Phase* we consider the jobs of the non-SC doors application. Jobs are ordered according to the sub-matrix $\widetilde{A}$ of the assignment matrix A. As jobs j_9 and j_{10} need temperature sensors, the only node usable for them is node n_3. Since these jobs have the least number of usable nodes, they are ordered first. All other jobs have the same number of usable nodes. The allocation compatibility sub-matrix is shown in Table 4.7 whereas Table 4.8 shows the sub-matrix $\widetilde{C}$ used for the communication job ordering heuristics. The communication heuristics is used to order all tied jobs. From Table 4.7, if we order the summation of rows according to the ascending order then jobs j_9 and j_{10} comes first in the job ordering and if we order the summation of columns according to the descending order then the node n_3 comes first in the node ordering which is the only usable node for j_9 and j_{10}. Jobs j_9 and j_{10} are still tied for first place and the job ordering becomes as follows: $j_9, j_{10}, j_7, j_{13}, j_{14}, j_8, j_{11}, j_{12}$ (with remaining ties broken by lowest index first).

A	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	$\sum$
n_1	1	1	0	0	1	1	1	1	**6**
n_2	1	1	0	0	1	1	1	1	**6**
n_3	1	1	1	1	1	1	1	1	**8**
n_4	1	1	0	0	1	1	1	1	**6**
$\sum$	**4**	**4**	**1**	**1**	**4**	**4**	**4**	**4**	

Table 4.7: Building matrix $\widetilde{A}$ for doors control application

Job list	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}
j_7	**0**	40	30	30	30	30	60	60
j_8	40	**0**	40	40	40	40	0	0
j_9	30	40	**0**	0	0	0	0	0
j_{10}	30	40	0	**0**	0	0	0	0
j_{11}	30	40	0	0	**0**	0	0	0
j_{12}	30	40	0	0	0	**0**	0	0
j_{13}	60	0	0	0	0	0	**0**	0
j_{14}	60	0	0	0	0	0	0	**0**

Table 4.8: The sub-matrix $\widetilde{C}$ used in Phase 3

Nodes are then ordered using the sub-matrix $\widetilde{A}$ (see Table 4.7). As we see that node n_3 is ordered first. It is the least constrained node, allowing all jobs to be mapped onto it. The rest of the nodes are tied and we consider the following order in the example: n_3, n_1, n_2, n_4 (with ties broken according to lower index first).

Jobs j_9 and j_{10} are mapped onto node n_3. This is the only feasible assignment, since this node has the required temperature sensor. During the assignment evaluation, both schedulability and resource constraints are checked. Job j_7 is selected to be assigned next. It cannot be assigned to node n_3 since this would violate schedulability $S_t \in h_C$. Consequently it is mapped onto the next node n_1. Jobs j_{13} and j_{14} will also be mapped onto node n_1 for the same reason. Due to violation of constraints (timing and memory), jobs j_8 and j_{11} are assigned to node n_2. Finally, job j_{12} is assigned to node n_4 since no other assignment is possible. The mapping of doors control application is shown in Table 4.9.

n_1	n_2	n_3	n_4
j_7, j_{13}, j_{14}	j_8, j_{11}	j_9, j_{10}	j_{12}

Table 4.9: Allocation of jobs from the doors application

Comments

The resulting mapping of both SC and non-SC applications are shown in Table 4.10 and in Figure 4.6 for the brake-by-wire and doors applications. SC jobs are shown in the row of SC core whereas non-SC jobs are shown

	n_1	n_2	n_3	n_4
SC core	j_{1a}, j_{2a}, j_5	j_{1b}, j_{2b}, j_6	j_{1c}, j_{2c}	j_3, j_4
non-SC core	j_7, j_{13}, j_{14}	j_8, j_{11}	j_9, j_{10}	j_{12}

Table 4.10: Resulting allocation of jobs

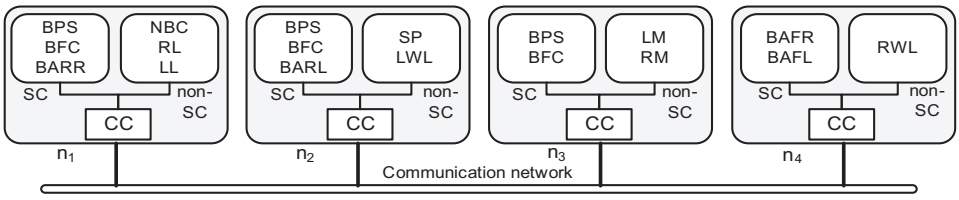

Figure 4.6: Resulting mapping – architectural view

in non-SC core. Furthermore it can be seen that the first node hosts three jobs from the SC application and three jobs from the non-SC application. The second node runs three jobs from the SC and two jobs from the non-SC. The third node runs two from the SC and two from the non-SC and the fourth node runs two jobs from SC and one job from the non-SC application. Figure 4.6 shows the architectural view of the resulting mapping. However in practice (e.g., aborad a car) this four nodes architecture may host more jobs until it can satisfy the underlying constraints. For demonstration purpose, we have shown the mapping of two small applications onto this four nodes dual core processors architecture. In order to tolerate faults at least 3 nodes are necessary for these applications since the highest critical job from the brake-by-wire application are replicated three times which must be assigned onto three distinct nodes.

Chapter 5

Multi Variable Optimization (MVO)

The design of heterogeneous embedded systems comprises of varied SC and non-SC functions, SW and HW components, several competing objectives/-variables including dependability/FT and RT among many others entailing the concurrent consideration of multiple constraints and competing variables in the design process. The design of such systems is performed under limited resources, i.e, they are resource constrained. An evolving design process is needed which can properly utilize the resources. On the other hand a systematic and efficient process is needed to tackle the complexity including the challenge of composite consideration of various design variables. Consequently, we develop a co-design optimization technique – the MVO approach, enabling the consideration of various design criteria and their quantification while satisfying multiple constraints and variables simultaneously instead of considering them on a discrete basis.

The *Research Question 4* [RQ4] is fully covered by this chapter. Before presenting the main aspects of the approach some essential issues related to the optimization process is discussed first. These include general optimization ideas, selection criteria/properties of the design variables, a discussion on quantifiers of the variables, *preferential independency* and *trade-off* analysis and the chosen set of design optimization variables. We provide *quantification* and *estimation* of each of the optimization variables. Given the premises on designing dependable RT systems we elaborate on *influence, scheduling length and bandwidth utilization* as co-design variables. The optimization process is employed to an existing metaheuristics algorithm called as *simulated annealing*. The effectiveness and the performance evaluation of the approach is presented in the next chapter.

5.1 Essential Issues in MVO

The importance of various design issues depends on the application model under consideration and on the number of variables. For instance we consider design and optimization of dependable RT embedded systems. A proper selection of variables is essential for the design optimization and in ascertaining trade-offs between variables. Therefore there are several points that need to be concerned about such as general optimization flow, pareto dominance, selection criteria of variables and evaluators for the variables. The crucial part is to define the exact set of variables for the target system design. In the subsequent sections we describe all these aspects.

5.1.1 General Optimization Ideas

A general structure found in most multi objective optimization literature of how to get an improved solution is stated below in Algorithm 5. This is often called as iterative improvement algorithm where a move is only accepted if the *candidate solution* is better (smaller the better for minimization and greater for maximization) than the current solution. However this process may often get stuck in a local optima. We depict a design optimization flow describing the steps in Figure 5.1. This structure is followed by the existing meta-heuristics like simulated annealing, tabu search and genetic algorithm which integrate various techniques for generating the candidate solution. These algorithms employ their own technique (as described in Section 2.1.4) to avoid getting stuck in local optima.

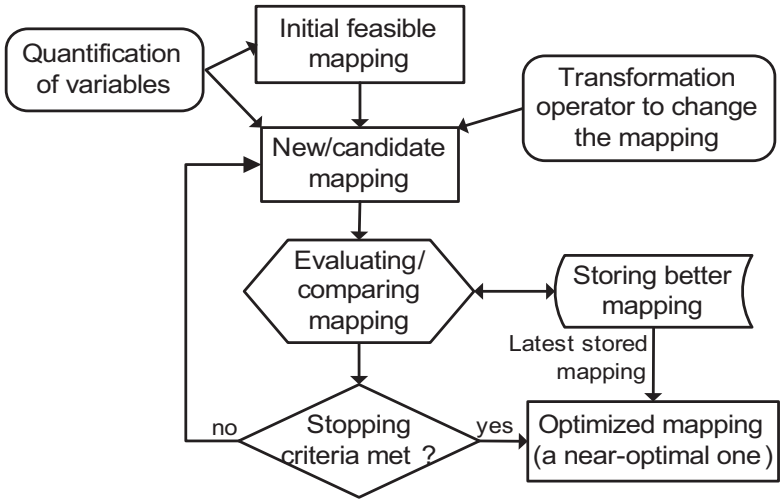

Figure 5.1: Design optimization flow

Algorithm 5 General optimization process

1: Generate a current mapping $f(x)$ and evaluate its merit (either minimization or maximization);
2: Generate candidate mapping $f(x')$ from $f(x)$ by applying transformation to the current mapping $f(x)$ or using the heuristics (guided or randomly);
3: **if** Evaluation of $f(x')$ is better than evaluation of $f(x)$ **then**
4: $f(x) = f(x')$; /*Current mapping is replaced by the candidate mapping.*/
5: **else**
6: The current mapping $f(x)$ is kept;
/*If $f(x') < f(x)$ for minimization and $f(x') > f(x)$ for maximization then the mapping $f(x')$ is treated as a better solution, if equal or even worse mapping is found then worse move like in tabu search or acceptance criteria is considered like in simulated annealing.*/
7: **end if**
8: **if** Stopping condition is met **then**
9: Finish;
10: **else**
11: Go to *Step 2*;
12: **end if**

The design optimization flow shown in Figure 5.1 starts with an initial feasible mapping. In order to find a better mapping a candidate mapping is generated and compared with the current mapping. A transformation operator (see details in Section 5.3.3) is employed to guide the move for creating a candidate mapping. The better mapping is stored and then always used as the current mapping. If the candidate mapping is better then the current mapping is replaced by the candidate mapping and stored as better ones. Whenever a new mapping is created the quantified values of all the considered variables is updated and used in every run of the optimization. If a stopping criteria is met for example the algorithm finishes a maximum number of *a priori* set iterations or a certain parameter like the temperature is reduced to an specific low value, e.g., in simulated annealing, then the optimization process is terminated. The final stored mapping is treated as the *optimal mapping* or it can be a set of *pareto optimal*.

Pareto Dominance

A multi objective optimization problem can be represented as finding a set of pareto optimal solutions where several objectives are treated separately instead of combining to a single function. The solutions are selected according to a guidance called dominance rule. An alternative design vector A_1 dominates another A_2 *iff* there exists one variable of A_1 which is strictly better than the corresponding variable of A_2, with all other variables of A_1 being

better than or equal to each corresponding variable of A_2. Suppose we have two distinct design vectors $A_1 = (q_1, q_2, \cdots, q_n)$ and $A_2 = (q_1', q_2', \cdots, q_n')$ containing the objective values of two alternatives for an n-objective minimization problem, therefore:

(α) A_1 strictly dominates A_2 *iff* $A_{1i} < A_{2i}$, for $i = 1, 2, \cdots, n$;

(β) A_1 dominates A_2 if $A_{1i} < A_{2i}$, for at least one i and $A_{1i} \leq A_{2i}$, for $i = 1, 2, \cdots, n$;

(γ) A_1 and A_2 are incomparable if neither A_1 dominates A_2 nor A_2 dominates A_1.

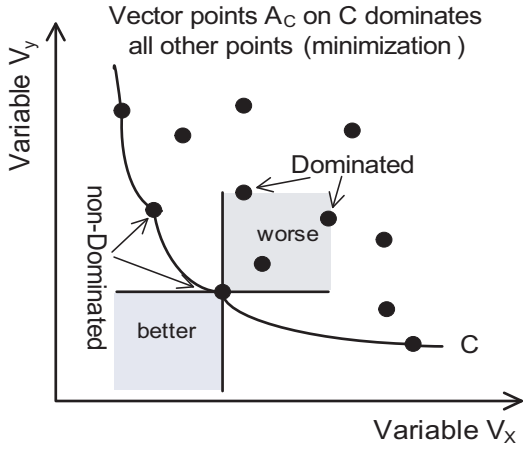

Figure 5.2: Dominance and effectiveness frontier

The first (α) and second (β) conditions can be applied directly to discard the dominated design alternatives by the aspiration vector or discard if dominated by each other among the alternatives. The dominated alternatives are the worse designs and do not satisfy the desired system requiremetns/aspiration values. Design alternatives which falls in the third (γ) condition remain in the decision process. When the vector of variables is represented as weighted sum then the decision is taken based on the output sum. The set of vectors which are not dominated is called the *effectiveness frontier*. These are depicted in Figure 5.2. The points of vector A_C on effectiveness frontier C, which also known as *Pareto optimal set*, dominates all other points in the graph. For minimization problem, points which lie on the upward/front direction of the curve C are the worse points and which lie on the downward direction are the better points.

5.1.2 Properties of Co-Design Variables

A consistent and complete set of design variables are needed to define so that the chosen variables are adequate in indicating the degree to which overall objectives are met. Generally in case of design with multiple objectives, a set of selected onjectives/variables should satisfy some properties [116]. Though there is no concrete way or any step-by-step procedure to select variables, following points should be kept in mind while considering them in the co-design and optimization process.

- It is essential to have a clear picture about adding variables so that they cover important aspects of the problem. It is helpful to ensure the completeness property of variable selection, i.e., whether the selected set of variables cover overall optimization criteria. For example, whether it covers the main design issues of FT-RT embedded systems.
- The variables should be operational meaning that the selected ones should be valuable.
- Avoid redundancy of variables, i.e., avoid selecting variables which is same in operation but different in names.
- The dimension of the metrics should be minimum. It is a trick to minimize the effort, time and cost of evaluating the variables and solving the problem.
- Considering one variable may affect one or more other variables, thus ascertain trade-offs between them.
- A small amount of dependency between variables can be ignored within a sufficient local region of the comparison space [148].
- When the number of variables increases, e.g., if more than 6 variables are needed to handle, then it should be possible to decompose them. As an example if the problem deal with seven variables, it should be possible to break the problem into two subproblems, one involving four and another involving three.

5.1.3 Quantifiers of Variables

To quantify the amount of variables in each design, we need evaluators for them. For instance, in case of communication variable, one possible evaluator for communication may be number of messages sent and received per execution or period time. A possible evaluator for fault containment would be to

take the inverse of the minimum influence value between two nodes which results in maximum fault containment. Some criteria (e.g., mathematical formulation) would help to quantify the specific variables value. Assuming n evaluators, one for each variable and applying n evaluators to a design results in an n-dimensional vector. More formally, denoting the i^{th} variable by v_i, the i^{th} variable evaluator by E_i, and the quantity of the i^{th} variable by q_i, then $E_i(v_i) = q_i$. Design solution is then represented as multi variable metrics $A = (q_1, q_2, ..., q_n)$, where A is the design alternative of n evaluators $E_1(v_1), E_2(v_2), ..., E_n(v_n)$. In Section 5.2, we quantify different variables for the design of dependable embedded systems.

5.1.4 Integration Trade-Offs and Preferential Independence

As the complex design involves multiple conflicting criteria, it is in fact true that no dominant alternative will exist that is better than all other alternatives in terms of all of these variables. This phenomena entails the trade-off analysis. *Trading-off* between variables means making the compromise between the variables such that gain in one variable against another variable and vice versa. More specifically, for example, let the aspiration alternative $A_a = (q_1, q_2, q_3, q_4)$ and a design alternative $A_i = (q_5, q_6, q_7, q_8)$. Assume also that $q_1 = q_5$, $q_2 = q_6$, $q_3 < q_7$ and $q_4 > q_8$. The above problem is then summarized as follows: Is *trading-off* some of q_4 for more of q_3 in A_a ok, such that A_i is obtained? If *true*, then design A_i is *good* since it is *not dominated* and is *as preferable* as A_a. Otherwise, A_i is rejected as a bad design. Another design A_j is then considered and the same process continued. If we know the importance of the variables *a priori* and if the variables are preferentially independent then the trade-off between variables can be resolved by giving the weight to the individual variable according to their importance.

While integrating SW functions, some trade-offs might be necessary. For example, it may be preferable to assign two critical processes onto different HW nodes, but that may not be possible since both have to be replicated, and the number of HW nodes are limited. Specifically, if the HW has four nodes and two critical processes need to be triplicated then two sets of these replicates must be mapped onto the same node. Other problems might include such as need for a resource that present on only one processor, or a very high communication load. This is the basis for considering integration tradeoff [79] i.e., *Is there a limit to the level of integration one should design for.*

Preferentially Independence

It is defined as the trading-off between two variables is independent on any other variables. If v_1, v_2, and v_3 are three variables then trade-offs between v_1, v_2 is independent of third variable v_3, similarly trade-off between v_3, v_2 is independent of the value of v_1. In general, variables are mutually preferentially independent. To better understand this concept, let us consider the following example: Assume there are n nodes in the system, with node n_1 having a high computation load, and all other nodes $n_2 \ldots n_k$ have low computation loads. Also, the processes running on n_1 are such that they have a high degree of influence, such that the mutual influence value between them is high. Any possible trade-off would entail reassigning some of the processes on n_1 to other nodes, thereby balancing the load in the system at the expense of having lesser fault-containment, since highly interacting processes will then be located on different nodes. This is irrespective of whether another node n_i has two high criticality processes running on it, i.e., high criticality value of the system. In this sense, the trade-off between fault containment and load balancing is irrespective of the criticality value. Hence, these variables are mutually preferentially independent. Observe that we do not imply that the variables are independent. However, intuitively, the mutual preferential independence implies that, for a given trade-off, instead of looking at all the variables at the same time, the system designer can focus only on the variables of interest.

5.1.5 The Chosen Co-Design Optimization Variables

The design optimization of a dependable RT embedded systems naturally entails the variable selection from the dependability, RT and resource perspectives. Respecting the embedded systems co-design criteria described in Section 3.1, we have chosen variables for the optimization process so that the quality of the design can be evaluated properly. The first criteria is to obtain dependability by design where our focus is in two folds: *(a)* first to provide a certain level of FT and then *(b)* to confine the propagation of errors between nodes and to increase the robustness of message delivery over the network. We then consider a variable from RT point of view. We also consider the variables from resource utilization point of view. Nevertheless, without loss of generality designer can select their own variables according to their system requirements/goal and can apply into our optimization framework. The variables considered in this thesis are listed as follows in decreasing order of importance and are quantified and modeled in the subsequent sections.

(i) **Influence ($\hat{I}_f$):** As our main objective is to achieve and optimize dependability by design we endeavor to minimize the influences between different modules. This variable refers to how well the errors are contained within a single node. Low influence values between nodes implies good error containment. Assigning highly interacting jobs on the same node reduce the error propagation probability across nodes, i.e., the error does not propagate but contained within one node. Since dependability is the prime design driver for SC embedded systems, the importance of fault/error containment is uppermost.

(ii) **Slack and scheduling length ($\hat{S}_l$):** It represents the total completion time of the system and how unloaded a processor is. This aspect of the design is considered since having some slack in the system later upgrades is possible. The slack from an schedule can be utilized both for FT and for power savings. Minimizing scheduling length is benefited from the resource usage perspective where more jobs can be executed satisfying their deadlines or future upgrade of job is possible.

(iii) **Communication ($\hat{B}_w$):** The amount of communication volume between nodes is represented by this variable. It is considered from a performance and resource utilization point of view which reduces the communication overhead and allows to use a low cost network.

5.2 Quantification of Optimization Variables

We present the *quantification* and modeling of a set of variables for the design of mixed critical RT embedded systems. This includes how to estimate/measure variables, and how to formulate them in terms of *function minimization*. The primary objective is to enhance dependability by design, where our focus is to *minimize influences*, i.e., to reduce the propagation of errors between nodes. The second and third considerations are the *scheduling length* and the *bandwidth utilization* respectively which are important in terms of performance and resource utilization and consequently lead to designs with lower cost. In the subsequent sections, we provide details of the considered variables used to quantify and optimize the design objectives. We extend the framework by adding two more variables in the design optimization. One is again from the dependability aspect which is *reliability* and the other is *power consumption*. We describe them in the extendability and adaptability Chapter 8.

5.2.1 Influence

We endeavor to minimize the influences across jobs as well as across nodes by design. *Influence* is the probability of error propagation from a source to a target. Error, which is the consequence of a fault can propagate from a source (s) to a target (t) if the source s interact with the target t. We have investigated two potential ways in which influences between s and t could take place. Below we state them and provide details of how we estimate influence.

Case 1: Errors occur in the source and propagate to the target via message passing or shared resources. If a job is affected by an error of the node it is running on, it might propagate errors to jobs on other nodes with which it communicates/interacts or shares a resource. Such influences risk the failure of multiple nodes and are undesirable.

Case 2: Messages sent over the network can be lost or erroneous due to transmission errors. Erroneous messages can propagate to different nodes and may cause unexpected behavior.

Estimating influences

The influences shown in Figure 5.3 consists of three phases of error propagation, namely:

(1) an error occurring in a module or in a communication link,

(2) propagation of the error to another module, and

(3) the propagating error causing a *cascaded error* in the target module.

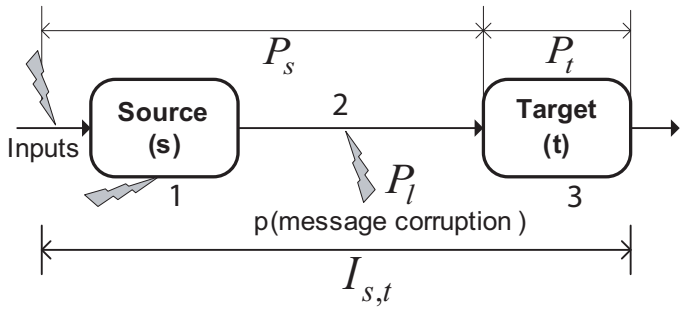

Figure 5.3: Error propagation

In order to measure influences, let us assume P_e is the probability of error propagation from s to t considering no corruption over the network and P_l is the probability of message corruption over the network. The probability of error propagation from a source (s) to a target (t) is denoted by $P_{s,t}$ and defined as follows:

$$\begin{aligned} P_{s,t} &= p\{error\ propagation | no\ corruption\ over\ the\ network\} * \\ &\quad p\{no\ corruption\ over\ the\ network\} \\ &= P_e \cdot (1 - P_l) \\ &= P_s \cdot P_t * (1 - P_l) \end{aligned} \tag{5.1}$$

Where,

$$\begin{aligned} P_e &= P_s \cdot P_t \\ P_s &= p\{error\ in\ output\ of\ s | error\ in\ input\ of\ s\} \\ P_t &= p\{error\ in\ state\ of\ t | error\ in\ input\ of\ t\ coming\ from\ s\} \\ P_l &= p\{message\ corruption | error\ on\ the\ communication\ link\} \end{aligned}$$

The probability that s outputs an error and sends it to the input of t is P_s. The probability that an error occurs in t due to the error received from s is P_t. The former indicates how often s allows errors to propagate out of s and the latter indicates how vulnerable t is to errors propagating from s. The probability of message corruption P_l can be seen as the unreliable message transmission over the network, which is calculated as follows. Assume that the failure rate of the communication link is λ_l.

$$P_l = 1 - exp(-\lambda_l \cdot \frac{b_{s,t}}{T}) \tag{5.2}$$

Where, $b_{s,t}$ is the size of the messages between s and t and T is the transmission speed. $exp(-\lambda_l \cdot \frac{b_{s,t}}{T})$ is the reliability factor due to message transmissions over the network, i.e, the probability that the messages are transmitted safely.

We further elaborate on different error probabilities. An error (e.g., a bit flip transient error) occurs at any inputs of s or generated from any other sources and may propagate to input of t, where an error may occur. The probability of an error in I_y the y^{th} input or the y^{th} source to propagate out of s is $P_s^{I_y}$ which is given by the following relation.

$$0 \leq P_s^{I_y} = p\{s | I_y\} \leq 1 \tag{5.3}$$

If there is more than one inputs or error sources, the above equation is generalized [131; 149] for calculating the error transmission probability P_s:

$$P_s = \sum_{y=1}^{Y} (p\{I_y\}/Y) * p\{s|I_y\} \tag{5.4}$$

Where Y is the number of inputs of s and $p\{I_y\}$ is the probability of occurring error in inputs or in any other sources.

The analytical model of calculating different error propagation probabilities across modules resemble the error detection probability in inputs and outputs of combinatorial and sequential circuits using random testing [150; 151].

Measuring by Fault Injection: We now describe an experimental estimation of influence using *fault injection*. The error propagation probability is estimated using the following procedure: *(a)* in each input I_y of s inject an error (one input at a time, i.e., no multiple errors), *(b)* observe the state and output signals of s and the state and outputs of t, and *(c)* use golden run comparison (i.e., comparing an injection run with a *golden* reference or the fault-free run) in order to detect when errors have occurred in either. Let the number of injection runs where errors in the output of s and in the state and output of t have been detected be denoted as $\eta_{err,s}$ and $\eta_{err,t}$ respectively. The total number of injection runs is denoted as η_{inj}. We then estimate the error probability as $P_s = \frac{\eta_{err,s}}{\eta_{inj}}$ and $P_t = \frac{\eta_{err,t}}{\eta_{inj}}$.

Overall System Level Influence

Considering both $P_{s,t}$ (comprises P_s and P_t) and P_l, the influence for a single error propagation path is calculated as follows:

$$I_{s,t} = P_{s,t} + P_l \tag{5.5}$$

$I^o_{s,t}$ is the overall influences between a set of jobs assigned together on a node and interacting jobs allocated on different nodes, which is expressed by the following equation:

$$\begin{aligned} I^o_{s,t} &= 1 - (1 - I^1_{s,t}) \cdot (1 - I^2_{s,t}) \cdots (1 - I^x_{s,t}) \\ I^o_{s,t} &= 1 - \prod_{\rho} (1 - I^\rho_{s,t}) \end{aligned} \tag{5.6}$$

where $\rho = 1, ..., x$ is the number of influences paths between two modules.

An Example: We consider the following example shown in Figure 5.4 where a job j_1 is assigned to a node n_1 and another two interacting jobs j_2 and j_3 are assigned onto n_2. The overall influence of node n_1 to n_2 will be:

$$I^o_{n_1,n_2} = 1 - [(1 - 0.4) \cdot (1 - 0.3)] = 0.42$$

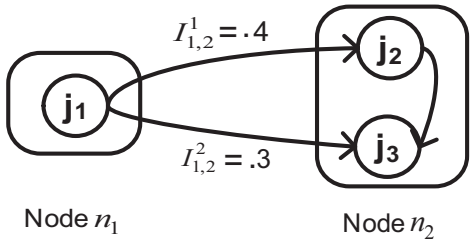

Figure 5.4: Combining influences

Influences are assumed to be zero for jobs which are assigned on the same node, e.g., the influence between j_2 and j_3. If all the three jobs could be assigned onto a single node then the error would contain within that node only. However it is not possible to assign all interacting jobs onto a single node due to imposed constraints. Also replicas need to be placed on different nodes which might have influences with other jobs. Hence, there will be jobs interacting across nodes. We strive to minimize these influences as much as possible for a mapping such that dependability is enhanced by design. Values for error occurrence probabilities can be obtained, for example, from field data or from system specification or by fault injection [131; 149]. The computation of the system level influence $\hat{I}_f$ is expressed as follows, which is then normalized, where k is the number of nodes:

$$\hat{I}_f = \sum_{i,j=1}^{k} I^o_{i,j} \tag{5.7}$$

5.2.2 Slack and Scheduling Length

This variable represents the total completion and communication time for a set of jobs on a node. As we use replication as the FT scheme, this results in more jobs needed to be scheduled which naturally incurs a overhead on scheduling. The goal is to minimize this overall scheduling length ($\hat{S}_l$) on a node satisfying precedence and deadline constraints. Minimizing scheduling length is important from the viewpoint of the uses of a set of processors,

since it leads to maximization of the processor utilization and the minimization of the maximum in-process time of the scheduled set of jobs. In every scheduling, gap may remain between two consecutive jobs executing on the same node due to precedence relations and communication cost. We define this gap as in-between slack (IBS). IBS can be used either reduce the frequency and/or supply voltage for energy savings or to schedule recoveries for higher performability which is defined as the probability of finishing the job correctly within its deadline in the presence of faults [152]. Therefore slack can be used for future upgrading of jobs and also for energy savings.

The scheduling length for a candidate mapping is calculated using the following equation:

$$\hat{S}_l \;=\; \forall k \; max \left[\sum_{i,j=1}^{n} (M_{i,k} \cdot CT_{i,k} + IBS_{i,j}) \right] \tag{5.8}$$

where, n is the number of jobs, $CT_{i,k}$ is the computation time of the i^{th} job in the k^{th} node, $IBS_{i,j} = EST_j - LET_i$, where i is the job executed before j on the same node, LET_i is the latest ending time of job i, and

$$M_{i,k} \;=\; \begin{cases} 1 & \text{if } j_i \text{ is assigned to } N_k \\ 0 & \text{otherwise.} \end{cases}$$

The scheduling length of a mapping can also be calculated by using the following expression:

$$\hat{S}_l = max_{\forall (i,k)} \{ft(j_i, n_k)\} \tag{5.9}$$

where, $ft(j_i, n_k)$ is the finish time of job j_i in node n_k which is equivalent to LET_i.

If the scheduling length is minimized then the CPU utilization, i.e., the computational workload per processor is maximized, which can be seen from the following equation termed as utilization factor (UF):

$$UF \;=\; \frac{S_l - \sum_{i,j=1}^{n} IBS_{i,j}}{S_l} \;=\; \frac{\sum_{i=1}^{n} (M_{i,k} * CT_{i,k})}{S_l} \tag{5.10}$$

Reducing the Length: There may be different techniques that one can attempt to reduce the scheduling length, however, violation of any hard constraints and the violation of achieving first objective/variable (a trade-off factor is weighed with influence) are undesirable. To reduce the scheduling length following acts can be performed with the transformation operator

(Γ): *(a)* Move a job which can be scheduled in a different node and select the allocation which result a reduced scheduling length, *(b)* If two jobs can be assigned to the same node then first assign with the lower EST which causes reduced scheduling length, *(c)* Scheduling length also depends on the *communication protocol* which can be seen as optimization of bus access scheme [137], and *(d)* Randomly move jobs and calculate the length of the scheduling after each move.

5.2.3 Bandwidth

In an integrated system design, jobs of different criticality and from different applications may be assigned onto a single node and jobs from a single application may be assigned onto different nodes. Therefore good utilization of shared communication links is necessary. Bandwidth utilization ($\hat{B}_w$) is the ratio between the total bandwidth required by the system and the available bandwidth of the network (B_T) defined as follows. The equation used here independent of a particular communication protocol.

$$\hat{B}_w = \sum_{i,j=1}^{k} b_{i,j}/B_T \tag{5.11}$$

where k is the number of nodes and $b_{i,j}$ is the total bandwidth requirements in terms of message size between nodes i and j. Minimizing this variable may allow for the use of a slower but cheaper data communications bus [67].

5.2.4 Example Describing the Metrics

Let us consider the same example which has already been presented in Figure 4.2 of Section 4.1.2. In this case, we try to show how different mappings can be created and their results with respect to system dependability and RT properties. We constitute the metrics for the considered optimization variables. All jobs in the application should finish their execution by $140ms$. The computation time CT, messages size $b_{i,j}$ between jobs as well as the influence values I_{ij} are shown in the Figure 5.5 (α).

Assume that each TDMA has equal slot length of 10ms and at least 2 messages each of size of 5bytes can be sent per slot considering the bus speed is equal to 1Kbytes/s. The largest message size is equal to the size of the slot which is 10bytes. We further assume that each node can only host any two jobs out of these four jobs. In Figure 5.5 (a), two interacting jobs j_1 and j_2 are assigned onto n_0 and j_3, j_4 are assigned onto n_1. The overall influences of the mapping, i.e., the probability of error propagation among nodes, is 0.2

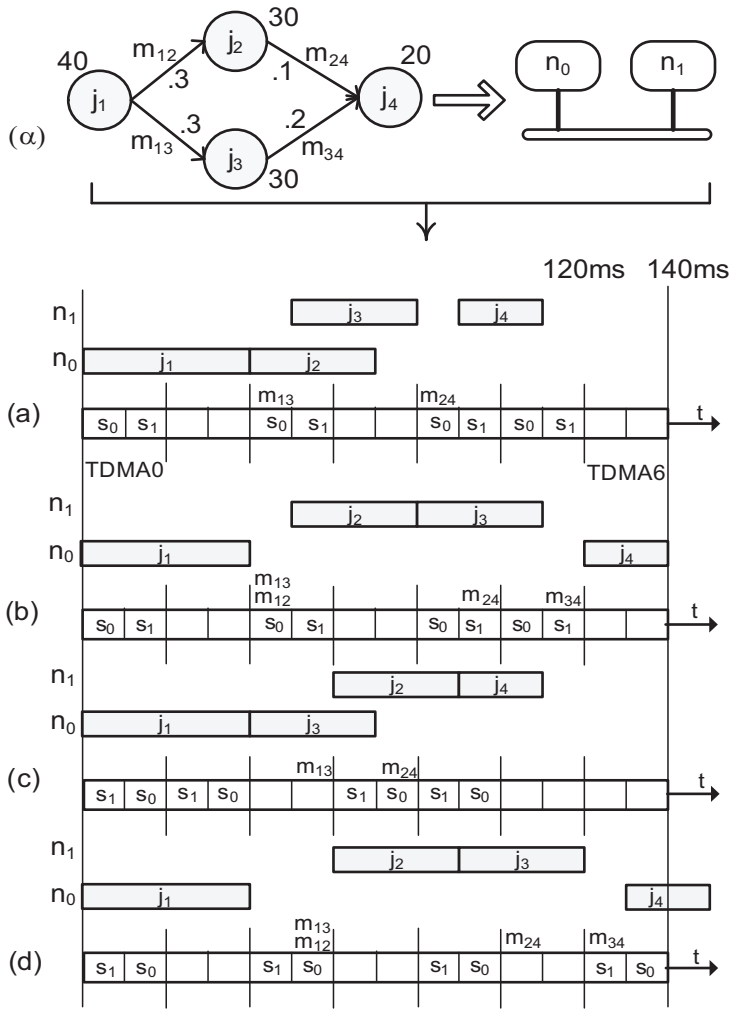

Figure 5.5: Quantification of variables for different mappings

calculated by using the Equation 5.6. The normalized scheduling length of this mapping is 0.78 and two messages are sent over the network. In Fig-

Influence	0.20
Scheduling length	0.78
Communication overhead	0.10

Table 5.1: Metrics of mapping configuration (a)

Influence	0.45
Scheduling length	1.0
Communication overhead	0.20

Table 5.2: Metrics of mapping configuration (b)

ure 5.9 (b), j_1, j_4 are assigned onto n_0 and j_2, j_3 are assigned onto n_1, where jobs are interacting indirectly. In this case the overall influence becomes 0.45,

Influence	0.22
Scheduling length	0.78
Communication overhead	0.10

Table 5.3: Metrics of mapping configuration (c)

Influence	0.45
Scheduling length	1.07
Communication overhead	0.20

Table 5.4: Metrics of mapping configuration (d)

i.e., the probability of error propagation is higher though it just maintains the deadline. The scheduling length is increased by $30ms$ and four messages are sent over the network. Since more messages sent over the network, the probability of propagation of error is higher between the two nodes and the possibility of messages get corrupted is also higher. Now, if the slot order is changed then this allocation violate the deadline constraint as shown in Figure 5.5 (d). The change of slot order does not affect on the first assignment policy. However the mapping with this slot ordering and the configuration of j_1, j_3 onto n_0 and j_2, j_4 onto n_1 is shown in Figure 5.5 (c). The quantification values of the variables for all the mappings (a)–(d) are shown in Tables 5.1, 5.2, 5.3 and 5.4. The mapping of Figure 5.5 (a) is considered to be the best mapping where the values for all variables are pareto non-dominated. We see that the value of the first variable is strictly better and others are equal or better compared to the (b)–(d) mapping configurations.

5.3 Our MVO–SA Approach

We apply an existing optimization algorithm within our integrated design framework. For this purpose, we have chosen simulated annealing (SA) [51]. SA is a metaheuristic based algorithm which converges to the global minima while solving an MVO problem, hence we termed this problem as *MVO-SA*. SA is a long established effective metaheuristic with an explicit schema for avoiding local minima [51; 83; 84; 52]. The reason for choosing SA is that it has been applied to a great variety of multiple objective optimization problems as well and is considered effective in general sense [153; 154; 155]. [83; 84] have shown the applicability of simulated annealing in RT tasks scheduling. Alternative approaches such as genetic algorithm [56], tabu search [55] were also investigated as options. However the global minima possibility with SA makes it attractive. A recent study has shown in [156] that SA works better than the evolutionary algorithms for many objectives test problems, i.e., for multi objective optimization problems.

Our overall optimization process is shown in Algorithm 6, which differs from the usual single objective SA. We have adapted SA for multiple ob-

jectives, which returns the best values of variables together with the best mapping found so far. In the following we list some features of SA, which led us to select it for the optimization.

- SA is applicable to a wide variety of complex, discrete and composite optimization problems which is a generic method and usually independent on the nature of the specific problem, thereof, we adapted it into our optimization problem,
- It has the technique to avoid local optima by accepting worse moves (so called *hill climbing*) and has the property of convergence to a global optimal solution,
- SA has the ability to scale to a large heterogeneous problem, and
- It is comparatively simple and easy to implement.

We now proceed this section by presenting the MVO function which is similar to a multiple objectives additive function [116; 157]. We then present the MVO-SA algorithm itself followed by the description of the different techniques used in the optimization process like exploring the search spaces by the *transformation operator* and comparing the value of different mapping.

5.3.1 The MVO Function

$MVO(v)$ is a function, which returns a natural number that corresponds to the overall quality of a given mapping. This section formulates the *MVO function* as a value function by employing the considered design variables. The value of the function depends on the quantified/estimated values of variables. The function can be formulated as a scalar-valued function $MVO(v)$ to each point v from a design space to an objective space, representing the system designer preferences such that $MVO(v)$ is minimized. When the condition of mutual *preferential independence* between variables holds, the value function can be represented as an additive form [116]. In order to break down the assessment of a complex multi variable value function into smaller problems that can be solved independently, additive value function is used, which is a powerful mechanism. Usually these problems are solved by scalarizing the overall function meaning that the problem is converted into one single or a family of single objective optimization problem. We construct the $MVO(v)$ function as a weighted sum of the function of each variable, which is a widely used method for this class of problems [116; 48; 157]. The value

of the function is determined by using the values of variables $\hat{I}_f, \hat{S}_l,, \hat{B}_w$, and the trade-off factors ψ_i, ψ_s and ψ_b.

$$MVO(v) = \psi_i \cdot \hat{I}_f + \psi_s \cdot \hat{S}_l + \psi_b \cdot \hat{B}_w \tag{5.12}$$

The individual values of the variables are represented in a matrix form: $M[v] \equiv M[\hat{I}_f, \hat{S}_l, \hat{B}_w]$. After performing a move, the function is denoted as $MVO(v^{'})$ and the matrix as $M[v^{'}]$.

5.3.2 The MVO Algorithm – Application of SA

We now employ the MVO approach in SA, which is a well-known solver to this class of problems. We present the *MVO-SA algorithm* that requires the following inputs. The output of the algorithm represents the optimized mapping of jobs onto nodes.

(i) The *workloads* (set of jobs and nodes including their properties) to create the initial mapping,

(ii) Design optimization variables $(\hat{I}_f, \hat{S}_l, \hat{B}_w)$, the $MVO(v)$ function and the trade-off factors (ψ_i, ψ_s, ψ_b),

(iii) The Γ operator to change the mapping (*neighborhood exploration* and guided local search), and

(iv) SA parameters - initial temperature T_{init}, the cooling schedule T_h and the cooling factor c_f for lowering the temperature.

At the starting point of the Algorithm 6, we initialize the *annealing parameters*, e.,g., by setting the heating temperature T_h and create the initial feasible mapping. The initial mapping either can be a feasible mapping or can be an infeasible mapping. However, in our case as we only consider the variables in structuring the MVO function, the initial mapping is assumed to be a feasible mapping. The feasibility of the mapping is maintained throughout the search by an external function call $\mathcal{C}(h_C)$, i.e., the best feasible mapping is sought. However when calling the function to check feasibility, i.e., to satisfy the hard constraints h_c, it has to be performed as efficiently as possible to reduce the run time of the overall algorithm. In this way we always guide the search in the direction of feasible solutions. The values of all variables are set in the MVO function (Equation 5.12) and $MVO(v)$ is computed in *Step* 4. In order to generate the candidate mapping, neighborhoods are explored in *Step* 6. We apply the *transformation operator (Γ)* to explore neighborhoods which is described in the next section. While applying this

Algorithm 6 The MVO algorithm

1: Initialization; heating temperature $T_h = T_{init}$;
2: Generate an initial mapping;
3: Create the matrix $M[v] \equiv M[\hat{I}_f, \hat{S}_l, \hat{B}_w]$ for this mapping;
4: Evaluate the initial mapping $MVO(v)$;
5: **repeat**
6: Explore neighborhood of the current mapping using Γ;
7: Generate candidate mapping;
8: Create matrix $M[v'] \equiv M[\hat{I'}_f, \hat{S'}_l, \hat{B'}_w]$;
9: Evaluate candidate mapping $MVO(v')$ for the new matrix;
10: Calculate $\delta v = MVO(v') - MVO(v)$;
11: **if** $\delta v < 0$ **then**
12: $M[v] = M[v']$ and $MVO(v) = MVO(v')$;
13: **else**
14: Calculate acceptance probability $a_p = e^{-\delta v/T_h}$ and generate $r = random[0, 1]$;
15: **if** $a_p \geq r$ **then**
16: $M[v] = M[v']$ and $MVO(v) = MVO(v')$
17: **else**
18: Restore the current mapping, i.e., keep $M[v]$ and $MVO(v)$;
19: **end if**
20: **end if**
21: Reduce the temperature T_h by using a cooling schedule $T_h = c_f \cdot T_{h-1}$;
22: **until** Some stopping criterion is met
23: **return** The best mapping $MVO(v)$ and the corresponding matrix $M[v]$

operator the feasibility of the mapping is checked. In *Step* 9, the candidate mapping $MVO(v')$ is evaluated in order to compare it with the current best mapping. For the first time run, the generated initial mapping is the current mapping. The current mapping is updated with the better mapping found in each iteration. If the difference δv is less than zero (minimization) then we choose the candidate mapping as the better mapping.

$$\delta v = MVO(v') - MVO(v)$$

If δv is greater or equal to zero, then the candidate mapping is accepted with a certain probability, called as the acceptance probability (a_p). One of the commonly used acceptance probability functions is [83; 84]:

$$a_p = e^{-\delta v/T_h}$$

The technique used by SA to not get stuck at a *local optima* is to accept some worse moves as the search progresses. For larger δv, i.e., when the candidate mapping is extremely undesirable, the probability of acceptance diminishes. If T_h is higher a_p gets higher, thus the initial temperature (T_{init})

is set to a sufficiently high value to accept the first few candidate mappings even they are worse. The value of a_p decreases as T_h decreases. If an acceptance criteria is met, the candidate mapping is chosen, otherwise the current mapping is restored and the process is continued. T_h is reduced according to a simple logarithmic cooling scheduling shown below which is the most commonly used one in the literature [48; 83].

$$T_h = c_f \cdot T_{h-1}, \quad 0 \leq c_f < 1, \quad T_0 = T_{init}$$

Where c_f is the cooling parameter that control the temperature. The value for c_f is usually chosen more than 0.90. *Reheating* of the temperature can also be applied if necessary. Reheating is the setting back the value of T_h to T_{init} after a certain number of iterations. We perform several iterations at the same T_h (so called *Metropolis Monte Carlo attempts* [51]) to cover a larger search space. The algorithm returns the best mapping found so far when the temperature is reduced to a certain value.

5.3.3 The Transformation Operator

The *transformation operator* Γ performs the moves to the current mapping in order to generate a candidate mapping, as shown in Algorithm 7, where $M(j_i, n_k)$ represents that the job j_i is assigned onto node n_k satisfying all constraints. Specifically, Γ generates the move to perform the *local search*, i.e., to explore the neighborhood and then the candidate mapping is created. In this way we traverse/explore the co-design space.

Algorithm 7 Transformation operator Γ

```
1: Function: Γ operation Γ(h_C)
2: repeat
3:    Select jobs to perform a move;
4:    Γ on M(j_i, n_k);
5:    Call the function C(h_C);
6:    if (h_C == true) then
7:       Perform Γ;
8:       return To the MVO Algorithm 6 at Step 7;
9:    else
10:      Do not perform Γ;
11:      Go to the Step 3 to select the next job;
12:   end if
13: until (h_C == true)
```

Three different kinds of move [48] are discussed below together with their respective size in terms of the number of allocated jobs (q) and the number

of assigned nodes (k). The operators Γ_r, Γ_s and Γ_i refer to the relocate, swap and interchange neighborhoods respectively.

(a) *Relocate* a job to a different node, $\Gamma_r = q(k-1)$,

(b) *Swap* the nodes between two jobs, $\Gamma_s = q(q-1)/2$, and

(c) *Interchange* the allocated jobs between two nodes, $\Gamma_i = k(k-1)/2$.

As already mentioned that while performing these moves, the feasibility of a mapping is maintained. After performing each move ($Step7$) the value of the MVO function is recalculated by using the Equation 5.12. The Algorithm 7 ($Step5-6$) checks and maintains the feasibility of the mapping satisfying $h_C = \{F_t, S_t, M_r\}$, where F_t, S_t, M_r are correspond to the fault tolerance, schedulability and memory resource consumption respectively. A move is accepted when it satisfies all the constraints. The $\mathcal{C}(h_C)$ function checks the constraints which is shown in Algorithm 4. Note that, during the optimization process we do not change the assignment of jobs which require binding functionality B_f, i.e., the jobs that associated with special resources.

5.3.4 Comparing the Mapping

After a successful move, the candidate mapping is evaluated and compared with the current stored mapping in *Step* 10 in Algorithm 6. Whether a mapping can be accepted or not, is also determined as follows.

(a) A jump from a feasible mapping to an infeasible never occurs, i.e., feasible mapping always wins even if the infeasible mapping is better than feasible mapping in terms of value,

(b) Two feasible mappings are compared based on their MVO function values. The lower value is better for minimization problem, and

(c) A jump from a lower value of MVO function to a higher function value may occur with a decreasing probability as the heating temperature T_h decreases.

A candidate feasible mapping can lead to three possibilities for the status of the optimization variables in a matrix such as:

(i) Improve all the variables in the matrix,

(ii) Improve some of the variables and some others deteriorate or remain same, and

(iii) Deteriorate or worsening all the variables in the metrics.

In the first case it is easier to decide the mapping immediately since the candidate mapping is better from every variables than the current one. The search should continue in the present direction. In case all variables deteriorate and the value of the MVO function gets higher, in SA the candidate mapping is evaluated by using some sort of acceptance function in *Step*13 of Algorithm 6. When some variables get worse and some get better, the mapping can be chosen as the priority based improvement of the variables. We attribute priority on choosing variables, for instance in our case first priority is to minimize the influences. If this variable is improving and others are worsening, we still consider the candidate mapping as a better mapping than the current mapping. And the search should continue in this direction as well. If the second and even lower priority variables are improving then mapping can be selected by using the acceptance criteria. In our algorithm, this is maintained by using the weighting factors in the aggregated MVO function.

Chapter 6

Evaluation

The co-design and optimization framework presented in the previous chapters is evaluated by performing a set of extensive experiments. We present experimental results to prove the effectiveness of the mapping heuristics and to evaluate the performance and quantitative gain of the optimization approach. The proposed heuristics are compared with existing base line approaches. We have performed extensive experiments which show the effectiveness (quality of the solution), performance (reducing the search space and finding a quick feasible solution) and robustness (consistent to perform the same mapping over many runs) of our mapping process. By using an independent tool [42] a validation of the output of the allocation is done to check the schedulability. An empirical evaluation including performance and effectiveness of the MVO approach employed in simulated annealing is provided. For a representative target study, our evaluation shows significant design improvements for the considered variables over contemporary analytical initial feasibility solutions. Results show significant quantitative gain in terms of *influences, scheduling length, bandwidth utilization, CPU utilization and FT overhead.*

We divide the design problem into multiple phases (ordering heuristics, allocation, scheduling, optimization and prototyping) and use different heuristics which provide a real basis for the optimization and reduce the complexity of the problem and save computation time significantly. We perform the evaluation based on an experimental setup considering randomly generated workloads. The *workload* is defined as a set of SC or non-SC jobs and nodes including their properties. After establishing this setup, we depict the performance and effectiveness of the approach by explaining the various experimental results. We show the *convergency* of the MVO algorithm and measure the quantitative gain comparative to initial contemporary solution. We also present different other aspects like *proper CPU utilization, load balancing and reduction of FT overhead.* At the end of this chapter a validation

and a comparative study of the approach is provided. This chapter addresses the *Research Question 5* [RQ5].

6.1 Experimental Setup

In order to evaluate the developed design concept and process at first we depict the experimental setup. We use randomly generated workloads consist of mixed-criticality sets of 10 to 200 jobs denoted as $J10$, $J20$,...,$J200$ respectively. All jobs, along with their replicas, are to be assigned in an optimized way onto the available nodes. Job properties are uniformly distributed within the following ranges and are given in Table 6.1: Replication factor $\in \{2, 3\}$, Interaction $\in [.04, .52]$, $EST \in [0, 80]$ ms, $CT \in [2, 20]$ ms, $D \in [14, 200]$ ms, Memory size $\in [4, 15]$ MB and Message size $\in [2, 120]$ bytes. The message transmission delay time (size of the exchanged messages divided

Replication factor	2-3
Influences	.04 to .52
EST	0 to 80
CT	2 to 20
D	14 to 200
Memory	4 to 15 MB
Messages	2 to 120 bytes

Table 6.1: Job properties

by the transmission speed of the link or need to be bounded by an existing communication protocol) between communicating jobs executing on different nodes are subtracted from the deadlines. The HW model comprises of varied number for nodes (from 2 to 10), which are connected to a communication link with a speed of $150kbps$. The number of nodes are varied according to the different experimental set up (workloads) which is mentioned with the respective experiment. Sensors and actuators are attached arbitrarily to nodes. The memory capacities of nodes have been chosen arbitrarily as $100, 150$ and 250 MB. Nodes n_2 and n_3 have sensors and n_5 and n_7 have actuators attached to them. We assume that for all the node processors the computing power and also the value of the failure rates are same unless we explicitly mention about these properties. Even though all these data were not derived from existing real-life examples, they are realistic with respect to current dependable RT embedded systems. The experiments have been carried out on a windows platform with Pentium 4 at 3Ghz with 2GB of memory. The implementation has been done in C/C++ programming language.

Supporting Data File

While performing the experiments different data files have been implemented which contain the necessary information for jobs, messages and for nodes. Accordingly we construct three different files. The content of the files were randomly generated. In case of job data file, the total number of randomly generated jobs are stored with each job ID, name, corresponding properties (dc_i, EST, CT, D, M_r) and the information whether a job needs sensor or actuator. In the message file, the communication and influences values are stored. Jobs are mentioned as sending and receiving jobs with the corresponding messages size and influence values (*sending job* $\rightarrow$ *messages size, influence values* $\rightarrow$ *receiving job*). The node data file contains the total number of nodes and their properties. The information includes node ID, name of the node, sensor/actuator attached to a node and the memory capacity of the node. All these information have been retrieved during the implementation of the heuristics and algorithms.

MVO–SA Parameters

In order to implement the SA based algorithm, it is necessary to set the annealing parameters. Mostly used in the literature [48; 51] and after investigating different runs of our algorithm with various workloads and configurations, we tune the MVO-SA parameters as follows: the value of the initial temperature (T_{init}) was set to 50000, the cooling factor (c_f) was set to 0.98, and the trade-off factors were $\psi_i = 1500$, $\psi_s = 20$ and $\psi_b = 500$ for influence, scheduling length and bandwidth respectively. In order to generate the candidate mapping we have performed two types of moves (random reallocation and swapping - 50% each of *Monte Carlo* iteration) at the same temperature to cover larger search space. Experiments showed that applying both types of moves together gives a better solution than only using a single type of move.

6.2 Performance Evaluation of the Mapping

We evaluate the performance and effectiveness of the heuristics based feasible mapping (the algorithm has been presented in Section 4.4) and describe the result in this section. The evaluation is carried out by considering both SC and non-SC applications and different applications patterns. The results are compared among different assignment techniques. The use of heuristics takes less number of *iterations* to find a feasible mapping compared to other techniques. As the mapping problem is NP complete, we do not show the

complexity of the algorithm, instead we show how we can get a feasible solution with less or no backtracking and possibly a good mapping.

6.2.1 Effectiveness

We are interested to show the performance (finding a feasible solution while reducing the complexity of the problem) of the heuristics. We observe that our multi-phase algorithm requires less or no backtracking to find a feasible solution. Several experiments have been carried out. First the assignment policy is applied with the job and node ordering-heuristics, we call it *Heuristic* solution. Second, we consider random selection of nodes which is the *Random* solution. Third, a *Thrashing* approach is considered which implements a different way of exploring nodes, where first node from the order is tried for every job to be assigned. If all constraints are satisfied, the selected job is assigned onto this node, otherwise next node is explored. According to the heuristics of considering *most constrained and conflicting jobs first*, high critical jobs are assigned in *Phase I* of the mapping Algorithm 3. When jobs are assigned in this phase, different nodes are selected for them in *Step* 6 of Algorithm 3. These considerations result a significant number of less iterations to find a feasible solution.

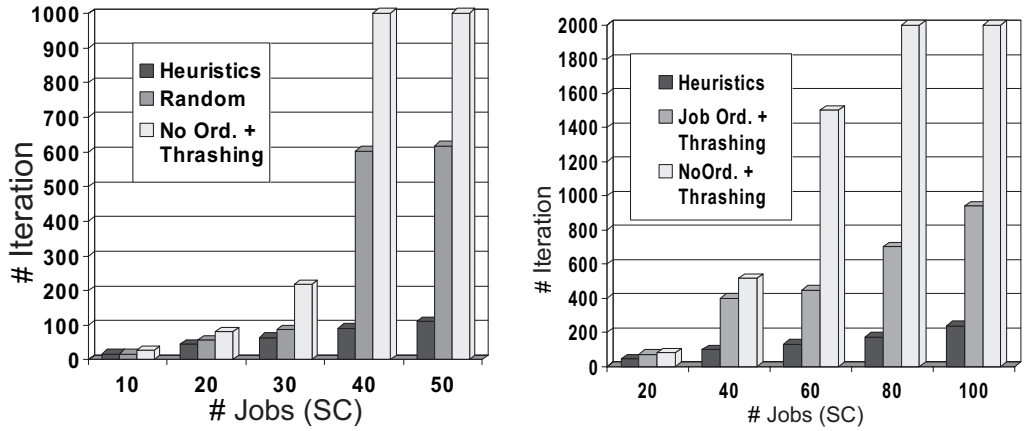

Figure 6.1: Performance of mapping heuristics (SC applications)

Figure 6.1 (a) shows the number of iterations needed for different assignment policies. Five nodes were chosen for this experiment. We observe that applying the mapping heuristics takes least number of iterations and hardly need backtrack to find a feasible solution. However this does not guarantee

that the backtracking is not needed at all to find feasible solutions. While performing the mapping for other workloads, the backtracking may be needed to create the feasible mapping. We have applied simple *swapping* (swap the nodes between two jobs) and reallocation (relocate a job to a different node) in the case backtrack was necessary. In case iteration bars touch the highest iteration line shown in Figure 6.1 (a) & (b), a feasible solution has not been found for that allocation policy despite of changing some assignments when backtrack was necessary. The reason might be that the search process got stuck to an infeasible design space. In Figure 6.1(b), the results found by heuristics process are compared with *job ordering + thrashing* and with *no job ordering + thrashing*. We observe that heuristics based solution requires least number of iterations to find a feasible mapping. In this set up (Figure 6.1(b)), the number of nodes were increased with the increasing number of jobs. 5 nodes were used for 20 and 40 jobs; $7, 8$ and 10 nodes were for $60, 80$ and 100 jobs respectively.

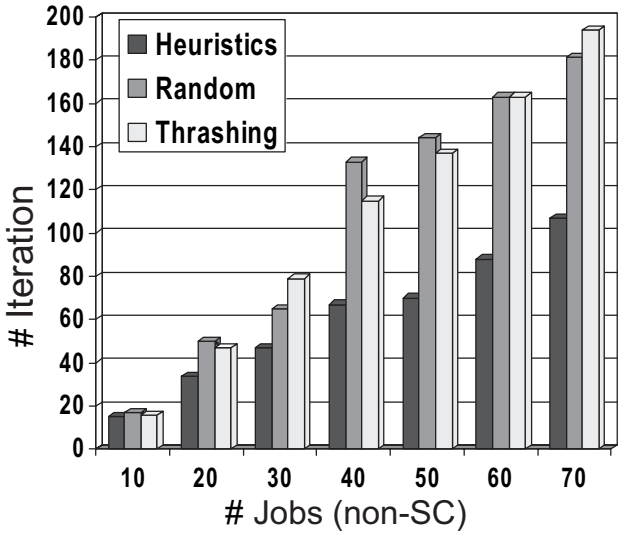

Figure 6.2: Performance of mapping heuristics (non-SC applications)

Figure 6.2 shows the result of the similar type of experiments, which has been performed for the non-SC jobs set. In this case the heuristics also works better than random or thrashing solution. Since there are no high critical jobs in non-SC applications, jobs are less constrained and only binding constraints do impact in the ordering heuristics. Four nodes were used for this set up.

Resource Utilization: We have performed experiments in order to compare the CPU and memory utilization of *heuristics* process with the *random*

and *thrashing* policy. We observe that the distribution of CPU and memory capacity by the heuristics approach is comparable with the random solution which is almost equally distributed among all the processors (see Figure 6.3 and 6.4). In case of thrashing, the load (computation and memory) among nodes are not properly distributed, i.e., are not properly load balanced. The measured utilization is based on the computation and memory available only for applications jobs. Resource consumption for middleware code and for other services are not included.

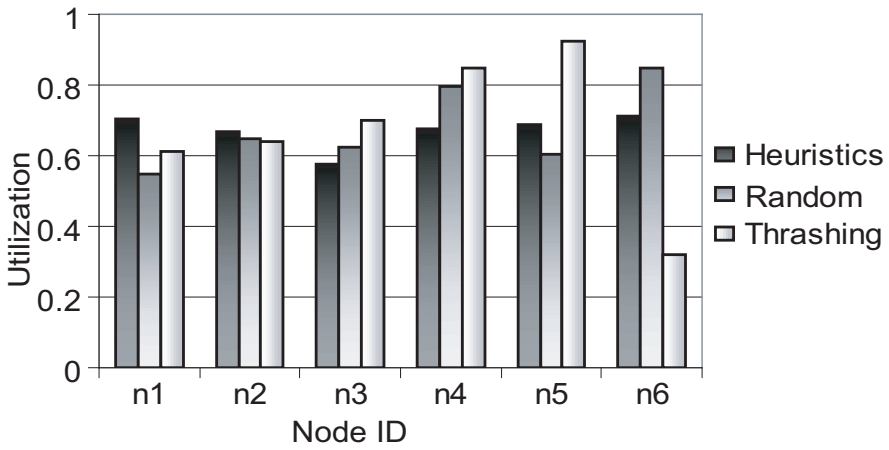

Figure 6.3: CPU utilization

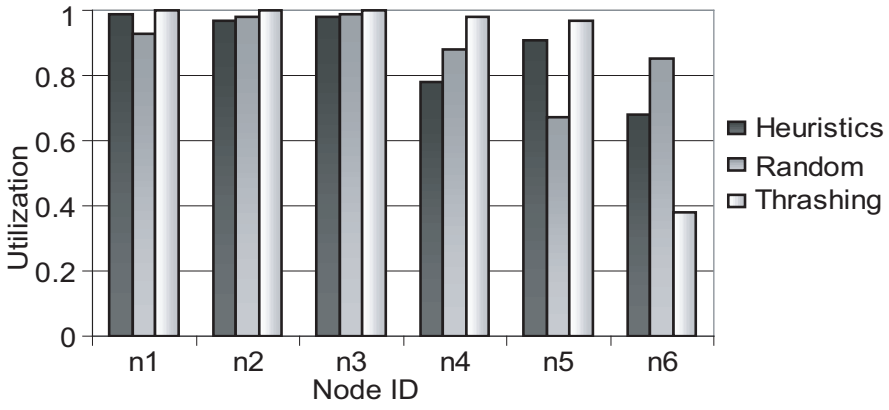

Figure 6.4: Memory utilization

6.2.2 Remarks

We have conducted similar experiments by using different application patterns. For example, deadline is set at the application level, therefore, all jobs within an application have the same deadline equal to the deadline of the application. The estimation of jobs properties are also varied, e.g., by changing the criticality degree, varying the computation time, deadline and messages size. We observe that our heuristics approach finds similar results to those discussed above. This shows the *applicability* and *robustness* of our algorithm on a wide area of applications. Furthermore, when we applied the communication heuristic, most of the communicating jobs are instantiated to allocate onto the same node. If we allow all these jobs to be assigned onto the same node satisfying other constraints, it may result poor load balancing. In order to tackle this we have applied *load balancing* technique (striving to assign the jobs equally among nodes while having the gain on communication heuristic) so that the loads are properly distributed among nodes.

6.3 Empirical Evaluation of the MVO-SA

We have performed extensive experiments in order to evaluate the MVO-SA and discuss the results in this section. For this experiment we have considered different set of jobs of $J40, J60$ and $J80$ and most of the cases an architecture of 8 nodes was assumed. First we show the convergency of the Algorithm 6 for different workloads to see whether there is an optimized design achieved per see. We then evaluate the performance of the approach by using various strategies such as computing the algorithm with different initial mappings and by adding extra resources to the platform. We depict the design performance profile as percentage gain in terms of the variables and the profile of the composite FT and RT gain.

6.3.1 Proof of Convergence and Effectiveness

As a part of the empirical evaluation of the optimization approach, we first observe the *convergence* of MVO-SA. Figure 6.5 shows that after a certain number of iterations with decreasing in temperature the MVO function reaches a minimum. At higher temperatures, more states have been visited by the *transformation operator* Γ to cover the large search space. We also see that for the same architecture configuration, in case of $J40$ the algorithm finds the convergence point earlier than $J60$ and $J80$ and takes less number of iterations to converge.

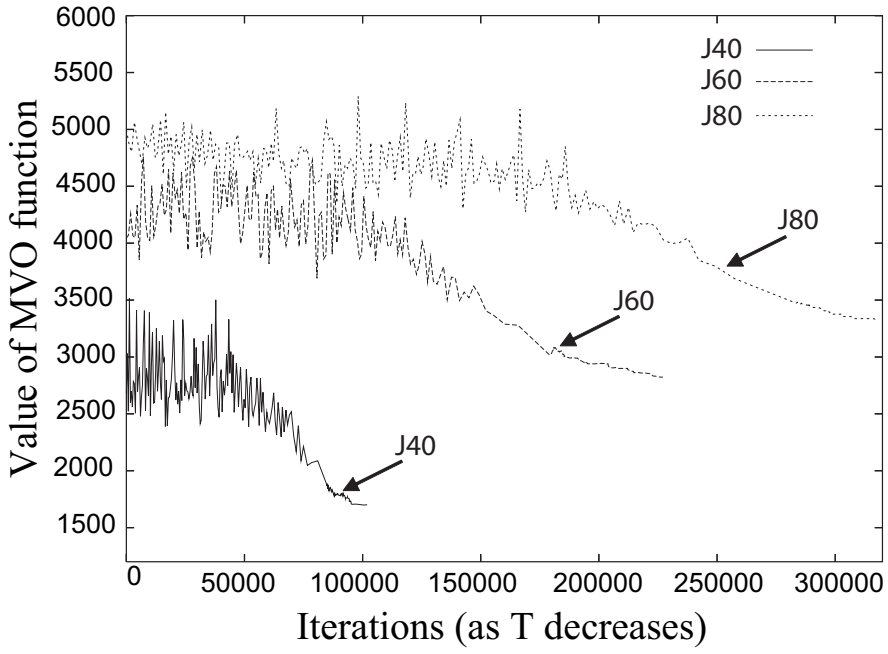

Figure 6.5: Showing the convergence

Given the proper selection of the parameters and the problem size, SA gives global solution by construction [51; 52]. Nevertheless, we have performed several experiments to evaluate if the MVO algorithm converges to a single point. Even though the algorithm is started with different feasible mappings ($Feas1$, $Feas2$ as shown in Figure 6.6), MVO-SA converges towards a solution every time. However the convergence points may differ negligibly, as shown in Figure 6.6 in case of $J60$ ($Feas1$, $Feas2$).

A good performance test of a mapping algorithm is to take a solvable problem and add resources [84], the algorithm should return a mapping no worse than the result of the original problem. We added two more nodes to the configuration of $J40$ and the resulted mapping displayed better performance. The convergence is shown in Figure 6.7 marked as $J40$ (10 nodes). To show the effectiveness of starting the optimization with a feasible mapping, we also have run the MVO-SA algorithm starting from an infeasible mapping. In this case the constraints and the objectives were considered at the same time during the optimization process. Though this can converge to an improved solution, it is slower than starting from a feasible solution (time for the creation of feasible mapping is included).

As mentioned before, the algorithm is started by using a feasible mapping which is generated at the same run either by using a heuristic or by a

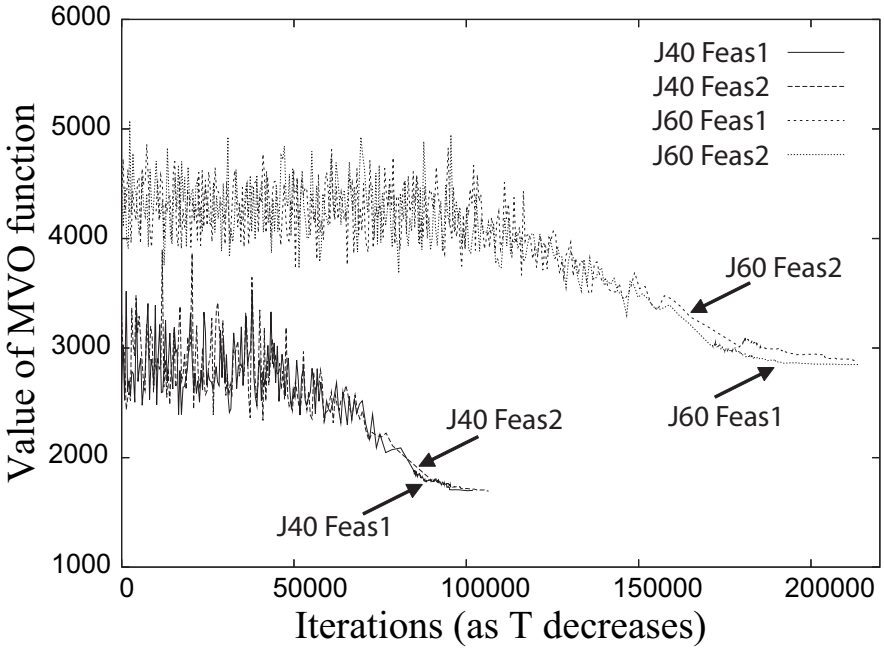

Figure 6.6: Performance evaluation of the MVO-SA

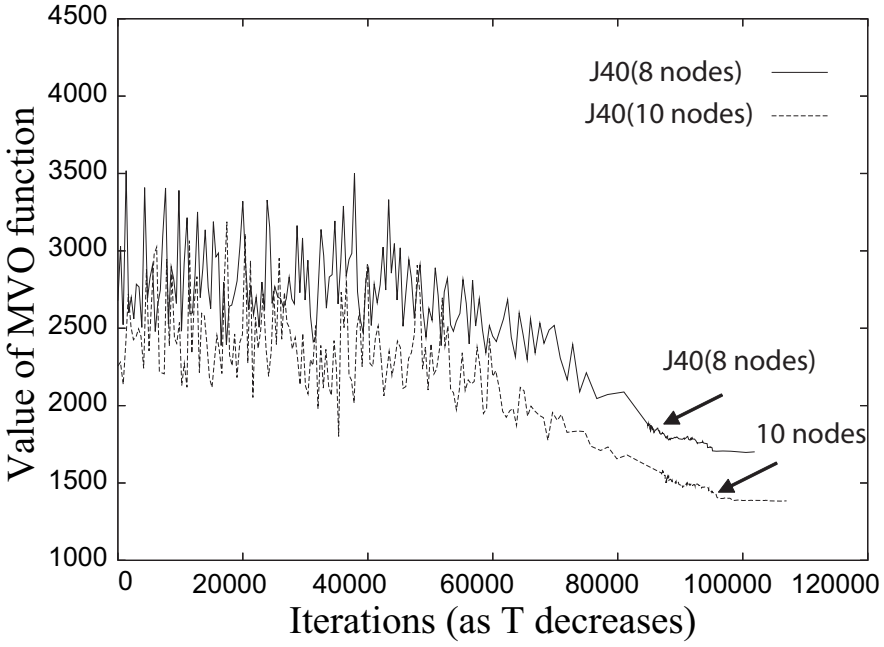

Figure 6.7: Effect of adding resources (laxation of constraints)

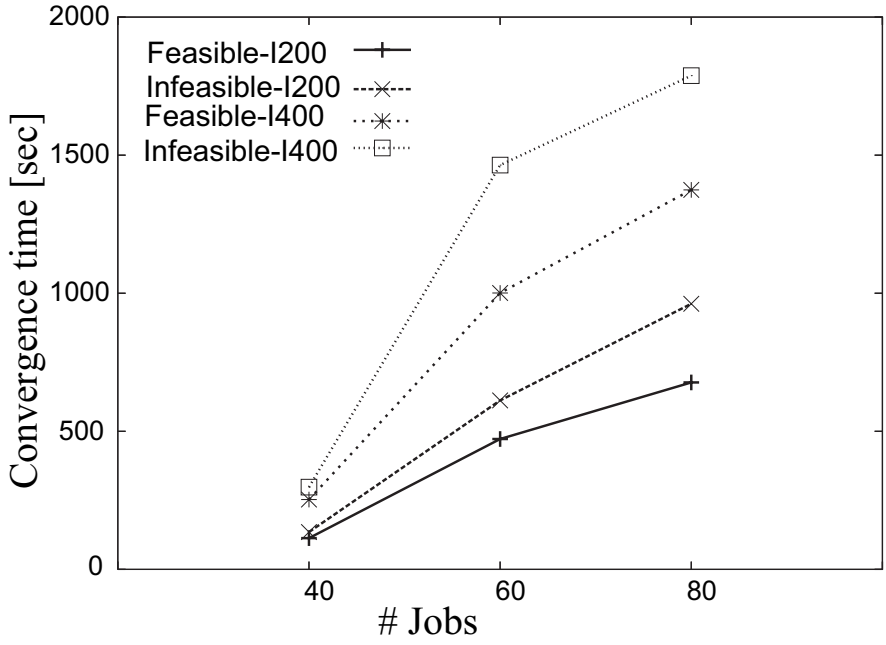

Figure 6.8: Run time comparison

random assignment. To show the effectiveness of this, we were also interested to evaluate the algorithm in case it starts from the scratch with an infeasible mapping. Though it can converge to an improved solution, cost a significant amount of time comparing to the former (when starts with a feasible mapping), as shown in Figure 6.8. This phenomena is also shown in Table 6.2 with *% speed up* compared with the optimization starting with a feasible solution. For larger workloads the difference will be more. For *Feasible I200* and *I400* graph the cost in time for creating the initial feasible solution is included. $I200$ and $I400$ are the maximum iterations used at the same temperature.

Workloads	Infeasible (s)	Feasible (s)	Speed up (%)
$J40$ – I200	135	113	20
$J40$ – I400	298	253	18
$J60$ – I200	612	472	30
$J60$ – I400	1464	1001	46
$J80$ – I200	962	677	42
$J80$ – I400	1788	1374	30

Table 6.2: Speeding up the convergency

In order to demonstrate the effectiveness of the framework, we also ran the algorithm considering large number of jobs set, e.g., $J200$. It took $44562s$ (around $12\frac{1}{2}h$) to find the convergence point with 1000 metropolis monte carlo iteration at the same temperature. Selecting the maximum iteration at the same temperature (the so called Metropolis Monte Carlo attempts) depends on the specific problem size such that the larger search region should cover within a judicious time as the simulation time increases linearly with increasing in iterations (see Figure 6.8). For each experiment, the maximum iteration is set accordingly.

6.3.2 Quantitative Gain

We are interested in evaluating the *quantitative gain* compared to a contemporary initial solution. As this gain depends on the value of the initial mapping, we have performed experiments using different initial feasible mappings. Table 6.3, 6.4 and 6.5 depict the mapping performance profile (M_{PF}) as percentage improvement for $J40$, $J60$ and $J80$ in terms of *(i)* $\hat{I}_f$, *(ii)* $\hat{S}_l$, *(iii)* $\hat{B}_w$ and also present the value of overall mapping $MVO(v)$ and the improvement. The values shown in these tables are the quantified values of the variables corresponding to the initial and final/optimized value of the mapping. For example, consider $J40$ and we see that the final mapping metrics $M[v^{opt}] \equiv M[.24, .32, .22]$ dominates the initial metrics $M[v^{init}] \equiv M[.42, .49, .29]$ according to the dominance rule described in Section 5.1.1. We consider minimization problem where lower value is better than the higher value.

	$\hat{I}_f$	$\hat{S}_l$	$\hat{B}_w$	$MVO(v)$
$M[v^{init}]$	0.42	0.49	0.29	2715.39
$M[v^{opt}]$	0.24	0.32	0.22	1746.89
$M_{PF}(\%)$	42.86	34.02	24.14	**35.67**

Table 6.3: Performance profile M_{PF} for 40 jobs

	$\hat{I}_f$	$\hat{S}_l$	$\hat{B}_w$	$MVO(v)$
$M[v^{init}]$	0.66	0.76	0.60	4304.95
$M[v^{opt}]$	0.32	0.54	0.40	2822.98
$M_{PF}(\%)$	51.52	29.1	33.33	**34.42**

Table 6.4: Performance profile M_{PF} for 60 jobs

	$\hat{I}_f$	$\hat{S}_l$	$\hat{B}_w$	$MVO(v)$
$M[v^{init}]$	0.68	0.92	0.70	5057.41
$M[v^{opt}]$	0.36	0.64	0.53	3358.78
$M_{PF}(\%)$	47.06	30.43	24.29	**33.59**

Table 6.5: Performance profile M_{PF} for 80 jobs

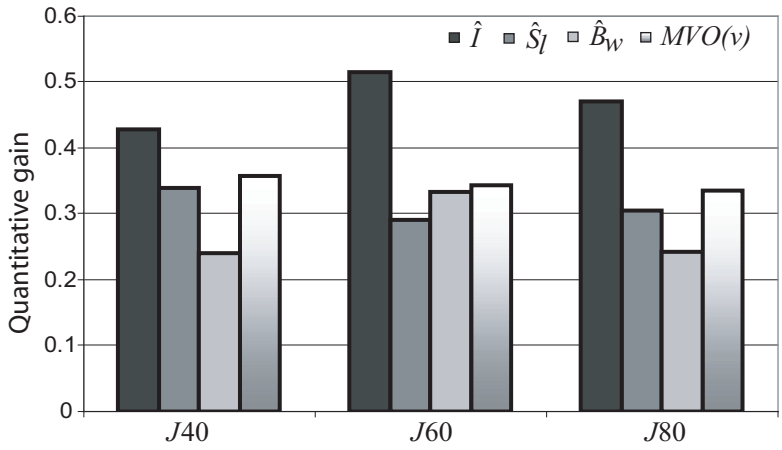

Figure 6.9: Mapping performance profile M_{PF}

In Figure 6.9 the performance profile M_{PF} is shown as relative gain with respect to the initial mapping which portray the overall M_{PF} for the workloads of $J40$, $J60$ and $J80$ in terms of $\hat{I}_f, \hat{S}_l$, $\hat{B}_w$ and $MVO(v)$. We observe that the gain is higher in case of $\hat{I}_f$, which ensures dependability driven design. In our case studies, on average, our approach has found 35% *better solutions* (composite gain), which leads to significantly better designs for dependable RT embedded systems.

6.3.3 CPU Utilization

Figure 6.10 shows the computation utilization by different node processors for jobs set $J40$, $J60$ and $J80$, which is about equally distributed among node CPUs, i.e., *a proper load balancing* is maintained by the approach. It is calculated by $UF = \frac{\sum_{i=1}^{n}(M_{i,k} \cdot CT_{i,k})}{S_l}$.

6.3.4 Reduction of FT Overhead

Due to the replication of jobs more number of jobs need to be mapped and scheduled which naturally incurs the overhead on resources and scheduling.

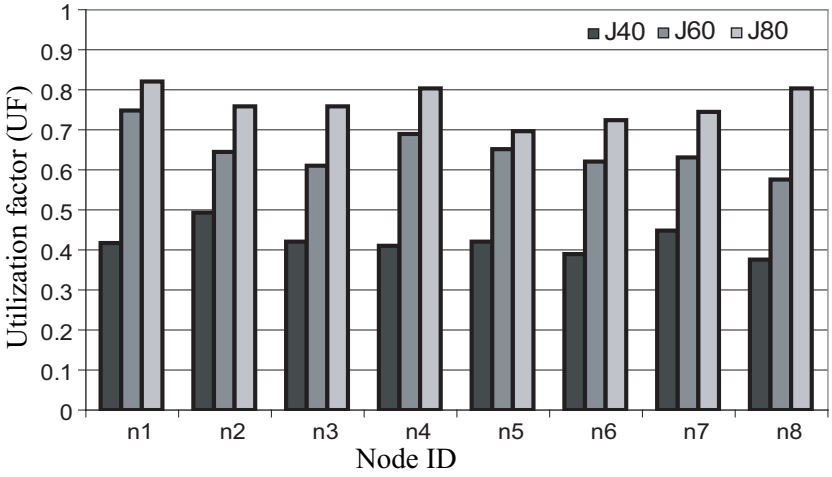

Figure 6.10: CPU utilization

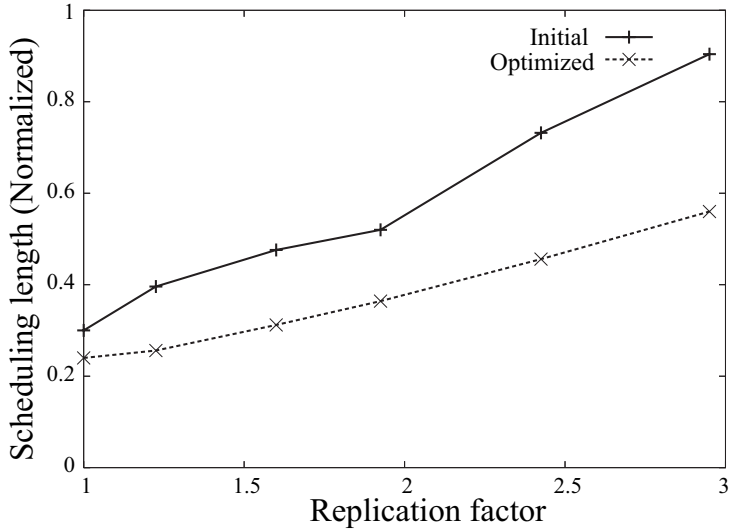

Figure 6.11: Reduction of replication overhead

However we try to assign them in an efficient way such that computational resources can be optimized. We observe the FT overhead (see Figure 6.11) both for initial and optimized mapping in terms of scheduling length. We varied the replication factor (Replication factor = # jobs after replication/# jobs) from 1 to 3. We have assumed that the highest criticality class of jobs need to be replicated three times. On average, the quantitative gain is 34.33%. Obviously, scheduling length has increased due to increasing the replication factor. Therefore, a design trade-off between RT properties and

the level of FT is necessary. The quantitative gain shows that the overhead is reduced significantly by the optimized mapping, which provides an FT design with reduced scheduling length.

6.4 Validation and Comparative Study

The result of the allocation process is validated using an independent tool [42]. The tool supports optimization-based allocation and scheduling of embedded time-triggered systems. Although, this tool uses mathematical optimization framework [114], it may lack of performance in case of large system models, so the tool and our solution proposed in this work can complement each other.

The optimization-based tool supports MVO in the scheduling phase, including criteria such as robustness, extensibility and throughput. The developer can select an appropriate composite objective function that delivers the needed combination of the target criteria. Our method uses heuristics-based search that delivers solutions quickly. This initial solution can either be validated (proving schedulability), or optimized using other tools. Our experiments show that the usage of allocation algorithm as part of the input to the optimization resulted in a significant performance gain. The result with less number of workloads shows that in case of optimization the presence of the initial feasible solution can help to reduce the search space by at least an order of magnitude. The experiments are done using multiple objectives function as described in [42]. The independent tool is used to perform the proof-of-correctness of the output of the allocation algorithm for different size of workloads. However we experienced some limitations while performing optimization in the tool using large workloads.

Both methods are complementing each other, because *(i)* the result of the heuristic approach can be validated and scheduled using the optimization-based method, and *(ii)* the basis solution can be used to improve the performance of the optimization.

Chapter 7

Prototype of the System Level Co-Design

In the previous chapters we have described the concept and methodology of various steps of dependability driven co-design and have evaluated the approach for embedded systems. We now develop the prototype of the proposed co-design methodology. In this regard, the concept and algorithms introduced in the earlier sections have been implemented in an open, extensible development environment. The task to develop a prototype representing the platform specific post integration information is a complex one. To ease the designer task, this should come with guidelines, methodology, patterns and rules. In this thesis the development of the prototype is performed by the guidance of the system level co-design adhering to the transformation based design principles such as model driven architecture [124] and platform based design [40]. The prototyping relies on developing concrete models (representing the design at particular stage) at different co-design stages and transforming the model from one form to another and from one level to another. Usually there are individual tools and technology which supports one or more development activities such as modeling the functionality, requirements specification, modeling the architecture, allocating jobs onto nodes, scheduling, performance analysis and optimization. However integration among these different technologies and tools in a convenient way are often lacking. Since different tools use different file formats and even different technologies, to make them work in a consistent manner is a difficult task and takes huge manual effort and time. In order to address this drawbacks our developed prototype integrates a set of tools and technologies on a single platform. In order to represent the system requirements and specification characteristics we use a platform independent specification model (PIM) at the higher abstraction level. By employing our co-design methodology into

a supporting tool set we transform this model to an architecture/platform dependent model which we call as platform specific post integration model (PSM). This model is then employed to control the task of deployment of a complete system onto a physical architecture.

This chapter starts with briefly describing the idea of transformation based design followed by the description of the overview and relevance of model and platform based design concepts. Different types of transformations are discussed to ease understanding the structure of the prototype. We present the architecture of the prototype and then describe different implementation steps of the prototype. Finally, the deployment and executable steps are depicted which enable us to run a complete system onto a target architecture. The supporting tools and technologies have been developed in conjunction with the DECOS (Dependable Embedded Components and Systems) project [145] partners.

7.1 Transformation Based Design

Using a transformational approach and adhering to model and platform based design principles, our aim is to convert/transform initial platform independent SW-C specifications into a platform specific post integration model (PSM) (Figure 7.1, below). The platform independent model (PIM) is created by specifying the functional, timeliness and dependability properties of application jobs, which is completely independent of platform specific programming details. We develop this model for varied criticality applications (detail of the application modeling is given in Section 7.2.1). For each application a single PIM is created. For SC application an SC PIM is created and for a non-SC application a non-SC PIM is created. On the other hand, the selected architecture/platform must support and host the functionalities while meeting the performance and dependability requirements. In [158], the HW architecture model is called as the cluster resource description (CRD) model. It is represented using a metamodel called hardware specification model (HSM) for capturing the resources of the architecture, e.g., computational resources, type of CPU, type and size of memory, communication resources, special purpose HW like sensors or actuators. The creation of CRD is also dealt with the resource composition to form the node describing the resource primitives. Adhering to model driven architecture (MDA), [158] has developed a tool based on this HSM metamodel which speeds up the modeling process. Primarily the PIM and HSM are created based on UML (unified modeling language) class diagrams and are then transformed into XMI format (XML Metadata Interchange) [126] such that they can be easily

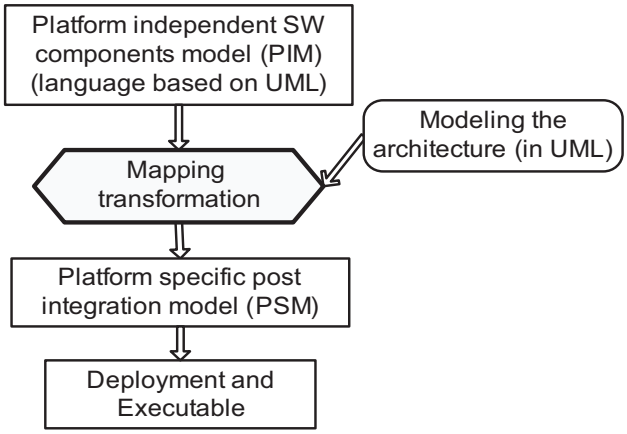

Figure 7.1: Transformation based design

used and integrated by other tools. At this stage mapping of SW-Cs onto HW-Cs is performed resulting a platform specific post integration model. Our aim is to provide an interactive (semi-automated) and iterative transformation based design (transforming PIM to PSM or so called PIM-to-PSM mapping). The PSM metamodel shown in Figure 7.1 treated as a candidate PSM which models the interconnection and instances of the mapping for example jobs are assigned on a node and communicating jobs may need to use the network but the decision has not yet been made such as the job has already been allocated onto that node.

A vigorous supporting tools and technologies in a VIATRA[1] based framework [159] is provided to perform the transformation. It is based on a metamodel driven infrastructure. Once all the defined transformations including allocation are done in VIATRA framework, the PSM of the target system is generated. The post integration phase is defined in the PSM, where system functionalities are already mapped onto the platform meeting various *requirements* and *objectives*.

7.1.1 MDD and PBD: Overview & Relevance

The emerging model driven development (MDD) and platform based design (PBD) initiatives address the model based design at a higher abstraction level. These two design concepts are used for system development which are based on models representing specification, design, etc. These technologies separate and relate the information of PIM and PSM depending on the

[1]VIATRA (VIsual Automated model TRAnsformations) is an open source model transformation tool.

level of the design. In order to develop an efficient transformation based design and adhering these methodologies, following three aspects have been characterized for this design approach:

(i) Modeling the application and jobs functional and extra-functional characteristics abstractly independent from the architectural implementation details,

(ii) Modeling the HW architecture such that it can support the functionalities defined in the first step, and

(iii) Transforming the mapping/integration such that the jobs functionalities are mapped onto available platform/architecture resources satisfying various design constraints resulting a PSM.

In PBD technology the third step is often termed as *meeting-in-the-middle* process. According to MDD, PSM is the level where system design meets implementation concepts and languages. According to that, the PSM is a view of a system from the platform specific viewpoint and is expressed in terms of the specification model of the target platform. There are various work which also use these concepts, e.g., by applying the MDD design environment and focusing on domain specific modeling language (DSML), [161] develops a tool suite which designs HW and SW based on the PICML (platform independent component modeling language) and ECSL (embedded control systems language) and integrates them onto an execution platform through generation of platform specific artifacts. In Section 7.2, we progressively formulate above three characteristics and create the prototype of the design process.

7.1.2 Model Transformation

The model transformation (MT) is the process of converting one model to another model which applies some rules like graph transformation [162]. In this context, transformation of the PIM to the PSM of the same system is performed. There exist many ways in which such a transformation can be done, e.g., manually, automatically or by computer assistance. In which way the transformation is done, it produces a model specific to a particular platform from a set of PIMs. Transforming PIM directly to PSM is a formidable task and has a large abstraction gap between them. It is necessary to generate intermediate models to bridge large abstraction gaps between models, instead of directly generating from the source to the target model. Therefore, in pursuit of developing the prototype we have classified different types of MTs to come up with the final PSM prototype. The MT is classified as the

intra model transformation which is done in the same abstraction level, the quality driven transformation which is done according to specific properties and the inter model transformation which is done between different level of abstraction models. These are described in the following:

Intra MT: This type of transformation is done on the same abstraction level, i.e., changes are propagated across models on the same level of abstraction, e.g., transformation which is done within PIM level, such as transforming the UML/XMI file created by rational rose or any other tools to the VIATRA compatible file format. Moreover, since the XMI files produced by the supporting tools may contain a lot of unnecessary information (such as graphical placement of model elements) which is then by transformation need to be simplified to be used in the prototype. The task of either creating the XML file or transforming the file into other format is tedious and error prone to do them manually. In this context it is done automatically by VIATRA model transformation tool.

Quality-driven MT: The key instigator of this transformation is particular quality attribute, for example, incorporating replication and job allocation into VIATRA can be categorized in this type where the transformation rules (e.g., rules that define the replication of jobs) plays a vital role. This type of transformation can be classified as intermediate MT as well. Transforming the allocation into VIATRA model space and transformation regarding interfacing the scheduler can be categorized into this class.

Inter MT: This type of transformation is done between models of different levels of abstraction, for example, when abstract application model is transferred to the technology specific model. The PIM-to-PSM mapping transformation is categorized into this type MT which is not directly performed in one step. An inter model transformation may contain more than one intra model and quality-driven transformations.

VIATRA – A Transformation Tool

The VIATRA2 (release version 2) model transformation engine addresses the challenges of model transformations and provides a complete system to define, develop, test, debug and maintain the model transformations and code generation. The PIM and CRD input models are transformed into the VIATRA model space and from where these models are used in other transformations for example in job replication and allocation. This way the

VIATRA model space is enriched by each transformation and at the end the prototype of the PSM is generated which enhances the deployment process of a system.

7.2 Architecture of the Prototype

The co-design steps are implemented on top of the open Eclipse tool integration framework. Eclipse is a Java-based, open-source extensible system which is currently one of the mostly used open platforms for tool development [160]. It supports specification of models using XMI, Java and XML schema. As the tool-chain works with multiple modeling languages and facilitates multiple model transformations, we use VIATRA2 a generic, open-source model transformation framework [159], which is an official Eclipse extension. VIATRA2 supports the reliable and simultaneous handling of multiple models, modeling languages and transformations as well as the code generation. The

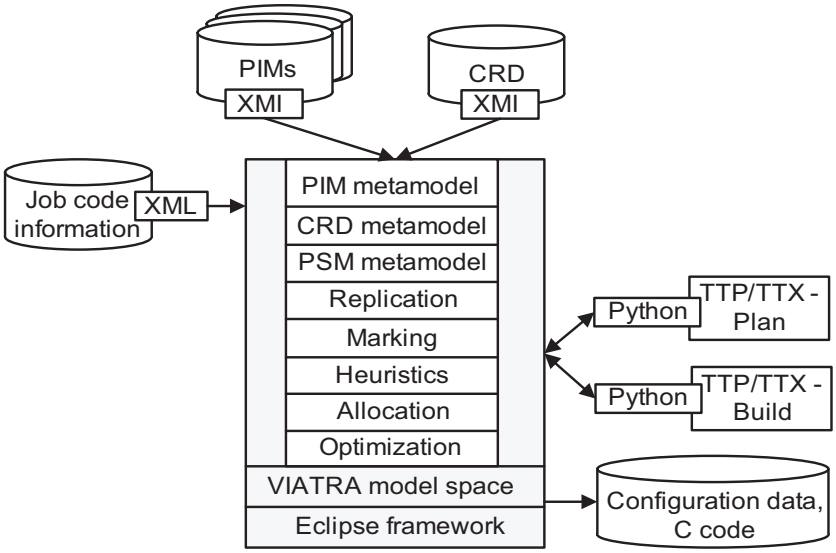

Figure 7.2: Prototype of the system level co-design

structure of the prototype implementation is shown in Figure 7.2, which is developed through appropriate techniques such as [173]:

- Supporting tool suite
- Implementing the allocation process
  - Transformation rules

  - Heuristics base

- Support for multiple model manipulation (transformations)
- Interactive basis – user interactive
- Smart graphical user interface

The inputs are the PIM models (each containing a single application) in XMI format, the CRD model (also in XMI format), and the job code information files (in XML). The PIM and CRD models have been transformed from the UML to the XMI format. These models are stored in the VIATRA model space as metamodels which are semantic frameworks for expressing models and their inter-operation. In order to improve usability of the tool, a custom user interface is implemented that hides the technical details of models and transformations and shows only the high level information. After the allocation and scheduling are done, all the necessary configuration files and source code for the system are generated. The SW is compiled and deployed using third party, system specific C compiler and SW download tools, e.g., TTP-Load [144]. The detail of these architectural steps including the supporting tool set is addressed in [163]. All the steps shown in Figure 7.2 are briefly described in the subsequent sections.

7.2.1 Modeling the Application and SW Development

The functional, performance and dependability requirements and properties are formally specified in PIM models and metamodels either by using a standard UML tool or by the dedicated domain specific editor [164]. To support the ease creation, the PIM model can also be developed by using the design patterns [165]. There are three packages that are created for different requirements of functional, performance and dependability. Jobs attributes are configured accordingly. The functional package contains different job characteristics, ports, information for messages, sensor/actuator jobs etc. The performance package contains the job information such as CT (WCET), deadline D, code size, etc. In the dependability package the degree of criticality is specified for the job properties, e.g., followed by the safety integrity level described in Section 2.1.2. Accordingly jobs are replicated in an early design phase and are executed onto distinct nodes. Not all PIMs create the dependability package meaning that for SC applications all these three packages are created and for non-SC applications creation of functional and performance packages is sufficed.

In parallel with the co-design process, the behavioral design of applications is need to be carried out which describe the definition of job behaviors. The applications or jobs codes are generated out of these behavioral steps. There are several commercial tools that can be used for this purpose, e.g., SCADE [166] and Matlab/Simulink [167]. SCADE is specially used for SC embedded applications (formally defined, deterministic) and two properties are interesting to get the information required for the prototype: *(i)* the code execution time is bounded and *(ii)* in order to meet DO178-B constraints [168], generated code from SCADE qualified code generator involves only a small subset of the ANSI-C language, with no fancy pointer arithmetic or such. This makes the estimation of WCET analysis reasonably applicable.

7.2.2 Transforming the Design Steps

In this section we describe the steps shown in Figure 7.2. In the first step the input models are imported to VIATRA model space. Importing the job code information (in XML) and sensor/actuator jobs replication (applying transformation rules) are then performed. The jobs code information is a feedback from the behavioral design process. A *marked PIM* is created by marking the platform specific information which are not yet given either in PIM or in CRD. This model contains the pre-allocation information like job placement definition and application interconnection definition. Job placement definition allows the designer to decide whether a job will be placed in a core node or will be in a node attached to a field bus. The application interconnection definition enables to mark the source and the target applications gateway messages for exchanging the inter-application information. The allocation compatibility matrix is defined in the marked PIM by marking the job compatibility, e.g., if a job needs a sensor/actuator it has to be allocated on a node attached with the sensor/actuator. This has been formulated as binding constraints. Replication is performed according to the degree of job criticality, in this case one-to-one job and resource mapping is done.

The heuristics and mapping algorithm are transformed into model space by applying the model transformation rules, i.e., by rules of graph transformation. An abstract view of how the mapping is considered in the transformation engine is shown in Figure 7.3. The PIM and CRD models (all are in XMI format) are used as inputs together with the estimation of job properties, e.g., the estimation of memory and code size, timeliness properties (EST, CT, and D).

The input and output file format of the message and job/partition scheduling is a python [172] file. The python file is generated from the model space with the allocation information. These are discussed in the next section. At

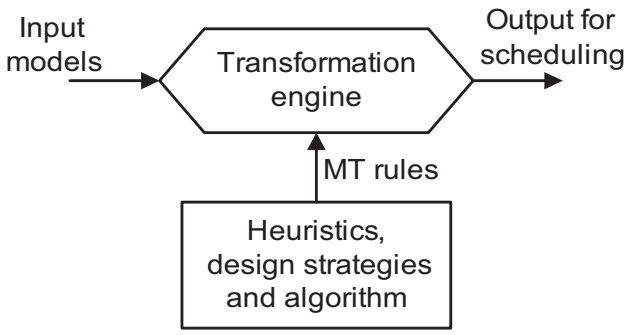

Figure 7.3: Allocation in model transformation

this point, all information is available that is needed to create message and job scheduling in the cluster. We decided to use third-party scheduling tools therefore the mapping process exports the data to the scheduler and then reads back the results. The final step of the whole process is the generation of application glue code and configuration files for the OS and for the communication controllers.

7.2.3 Scheduling Tool Support

A scheduling tool is necessary to provide a means of ease analyzing the timing constraints of the applications whether the constraints are met and assess that the deadlines are satisfied. Various tools exist which provide the scheduling of jobs/tasks, such as RapidRMA [169], VEST (Virginia Embedded Systems Toolkit) [170] and AIRES ToolKit [171]. These tools mainly use the rate monotonic analysis (RMA) for uniprocessor scheduling. Nevertheless, as mentioned in Section 4.1.3, TTP/TTX-Plan and TTP/TTX-Build are exploited in our approach as an example tools set for scheduling SC applications on a distributed architecture. These tools implicitly assess a feasibility test to check whether a given system assignment is schedulable and perform the scheduling and generate the necessary configuration files.

The input format of both tools is a Python script file. The input data has to be defined in form of a Python program that manipulates the internal database of the tool, creating all necessary elements before scheduling (cluster, nodes, jobs, messages). The mapping tool generates the Python scripts from the actual PSM, and invokes the tools. At first TTP/TTX-Plan creates the message schedule and communication configuration for the cluster. In the second step the node-level job scheduling is done by TTP/TTX-Build. This tool (also driven by an automatically generated Python script) loads the communication schedule and generates the node level schedule and the con-

figuration files of the operating system. Results from the scheduling process are the configuration of the communication controllers, the configuration of the node operating systems and stubs for the jobs. In order to generate middleware configuration code, a custom code generator is invoked that emits the code required to bind the operating system stubs with the behavioral code generated e.g., by SCADE. This code generator also configures the middleware modules that are necessary for legacy support.

7.2.4 PSM Prototype

The result of the mapping is the platform specific post integration model of the system that contains all the information about the SW and the HW components, and about the mapped jobs onto nodes. The model for the PSM is created and stored in the native XML format of VIATRA which provides the foundation of the mapping tool. The model does not need to be exported, as all transformations and marking steps are executed in the VIATRA framework. The creation of the PSM prototype is elaborated in [173].

7.3 Deployment and Executable

The generated PSM controls the deployment task of the system development process where all the source files (application code – generated from behavioral design either from a tool or manually coded, middleware code, configuration code for the OS) are compiled and linked into object files and executable files. Finally, in order to run a complete system these files are downloaded

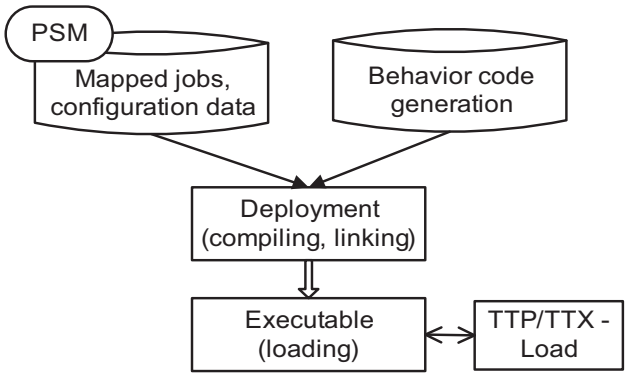

Figure 7.4: Deployment process

onto the target platform, e.g., by using the TTP/TTX-Load [144].

The deployment process to an actual target platform is divided into six steps as shown in Figure 7.5, which are controlled by a *Makefile* and are listed below [174].

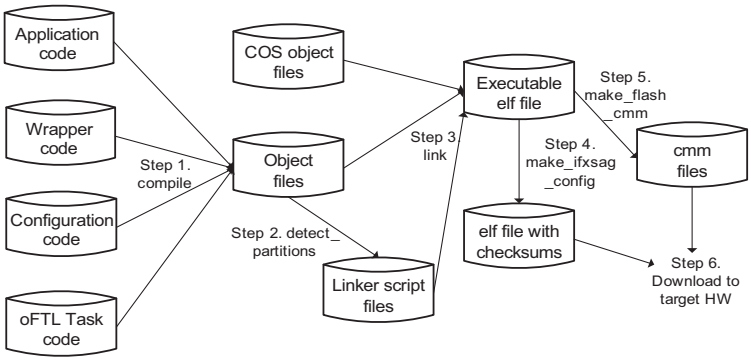

Figure 7.5: Deployment steps [174]

- All the C source files (application code, wrapper code, oFTL (optimized fault tolerance layer) tasks, Core-OS configuration code) [174] generated from the tool set are compiled into object files using the TASKING TriCore C compiler [175] in the first step.
- The second step is to detect partitions and generates linker scripts files which are crucial for memory protection and therefore for partitioning.
- In the *linking* step the generated linker script files are used to link the object files with the Core-OS.
- The CRC checksums of configuration data are pre-computed and written into the elf file using the Infineon SAG tool [176]. The Signature Analysis Generator (SAG) Tool generates the CRC checksum for the data in a linked ELF (Executable and Linking Format) format file before downloading.
- The last action triggered by the *Makefile* is the generation of a *cmm batch* file.
- Lauterbach TRACE32 [177] uses the generated batch file to download the *ELF* file to the target platform.

Chapter 8

Extendability and Adaptability

We have developed and validated the framework for an architecture which consists of node processors of equal speed and failure rate. We describe the extendability and adaptability of our approach in an architecture which comprises of node processors having different failure rates and speeds/frequencies. While performing the integration/mapping, it is necessary to use such an architecture efficiently. We describe how the framework can be extended to reliability measure and power optimization. Consequently we provide the quantification and estimation of these variables. Replication based FT scheme assures a certain level of dependability. We provide a measure of such level in terms of reliability. This variable can be improved during the mapping process when different failure rates of nodes are considered.

To explain the extendability and adaptability of the approach we assume that errors in node processors occur at a different rate of $\lambda_{n_1}, \lambda_{n_2}, \ldots, \lambda_{n_k}$ and errors in the communication links occur at a rate of λ_l. We also assume that node processors can have different speed/frequency $(f_{n1}, f_{n2}, \ldots, f_{nk})$. Therefore the computation time for a job with the same degree of complexity[1] will be varied on different nodes. The processors should be able to run in a different frequency level with different voltages.

In this chapter first we describe the adaptability of the mapping Algorithm 3 (presented in Section 4.4) on a heterogeneous architecture and illustrates the concept by using an example. We show the quality of the mapping in terms of the previously considered variables (influence, scheduling length and bandwidth utilization). We then discuss the extendability and adaptability of the optimization process by additional variables. In this regard, the quantification and modeling of reliability and power consumption are depicted. These variables either can be included with the variables presented in

[1]Complexity in terms of functionality – a more complex job is with more functionalities and requires long CT

the previous chapters or they can be considered differently in the optimization process. However this may increase the complexity of solving the problem. The co-design framework can be configured with the needed/required number of variables (either 3, 4 or 5) according to the importance of variables defined by the system designer.

8.1 Applicability on a Heterogeneous Architecture

We explain the adaptability of the mapping algorithm presented in Section 4.4 onto a heterogeneous (in terms of computation and failure rates) architecture/platform. A heterogeneous architecture may consist of node processors of different speeds (f_{nk}) and of different failure rates (λ_{nk}). Therefore each job may have different CTs on different nodes and also the jobs failure probability can be different. A processor with less failure rate is obviously more reliable and more jobs are assigned onto this node so that the system reliability is maximized. On the other hand a high speed processor can execute a job faster than a low speed processor. These may overload some processors while other processors are less utilized. However the high speed processors may have higher failure rates. Thus a better trade-off is necessary. In order to tackle this, heuristics like in [178] can be utilized while applying the retrospective technique in *Step 7* of Algorithm 3. The technique is *to allocate jobs onto a processor according to the product of the failure rate and the total time (T_t) required by the instantiated job and already assigned jobs on that processor. Assign the selected job on a processor where the product $\lambda_{nk} \cdot T_t$ is minimum.* This heuristic ($HetHeus$) provides a better trade-off between reliability and schedulability of the mapping. Moreover, if a job needs a certain level of reliability and only a specific processor can provide it then this job should be assigned on that processor or if a job needs high CT then it has to be assigned onto a high speed processor. These types of requirements can be taken into account in the allocation compatibility matrix A described in Section 4.2.

Illustration on a Heterogeneous Architecture

Using an automotive example, we demonstrate the applicability of the mapping process to an architecture consists of processors of different computing power and failure rates. For this illustration, we consider a HW architecture consisting of 4 nodes. Let us assume the speed of node processors n_0, n_1, n_2 and n_3 are $1 (for 125MHz), 1.25$ $(i.e, 156MHz), 1.5$ and

1.25 unit respectively. Therefore a job assigned on node n_1 requires 1/1.25 times less time to finish the execution. The failure rates (λ_{nk}) per hour are $1 \times 10^{-5}, 1.5 \times 10^{-5}, 1.5 \times 10^{-5}$ and 1.75×10^{-5} for node processors n_0, n_1, n_2 and n_3 respectively. The slots s_0, s_1, s_2 and s_3 are statically assigned with nodes n_0, n_1, n_2 and n_3 respectively. Maximum two messages of size 25bytes can be sent from each slot.

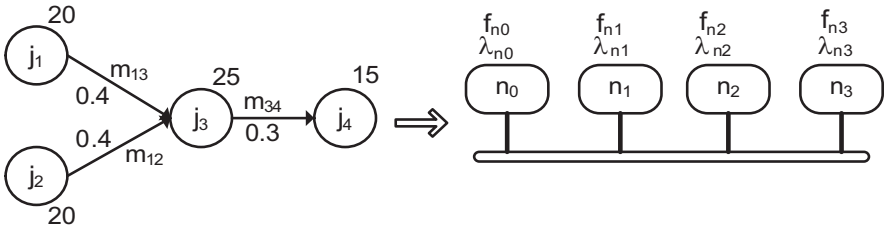

Figure 8.1: Application and heterogeneous architecture

We use the same application of brake-force control (BFC) as explained in Section 2.3.4 and assume the following job properties. Replication factor: 2, 2, 2 and 3; EST : 0, 0, 20 and $45ms$; CT : 20, 20, 25 and $15ms$ for jobs j_1, j_2, j_3 and j_4 respectively. The mentioned jobs CTs are for the processors of having relative speed of 1 unit (for $125MHz$). The values of CTs are updated according to the speed of the node processor where the jobs are assigned. Jobs j_1, j_2, j_3 and j_4 are correspond to speed sensor, distance control, brake force control and brake actuator job respectively. The influence value (probability of error propagation) between j_1, j_3 and between j_2, j_3 is assumed as 0.40 and between j_3, j_4 is 0.30. The message size between each pair of job is 25bytes. Jobs j_{1a} and j_{1b} are the replicas of j_1 and similar type of notation is used for all other replicas. All jobs from this application have to finish their execution by the deadline equal to the period of $150ms$. The chosen FT scheme tolerates one fault (either transient or permanent). The application and the architecture model is shown in Figure 8.1.

All jobs along with the replicas are assigned onto available nodes. Two mapping configurations shown in Figure 8.2 are illustrated as follows. For the first mapping (Figure 8.2 (a)) jobs are assigned without the guidance of the heuristics. For the other configuration (Figure 8.2 (b)), according to the ordering heuristics we first assign sensor jobs j_1 and j_2. For the remaining jobs, we then apply the $HetHeus$ in Step 7 of Algorithm 3. As all jobs are high critical only $Phase\ I$ is executed. Let us consider the case of assigning j_{3a}. We calculate the value of the product $\lambda_k \cdot T_t$ for nodes n_0, n_1, n_2 and n_3 which are $(20 + 25) \times 1 = 45$[2], $(16 + 20) \times 1.5 = 54, (14 + 17) \times 1.5 = 46.5$

[2]For simplicity 10^{-5} is discarded from the value of λ_{nk}.

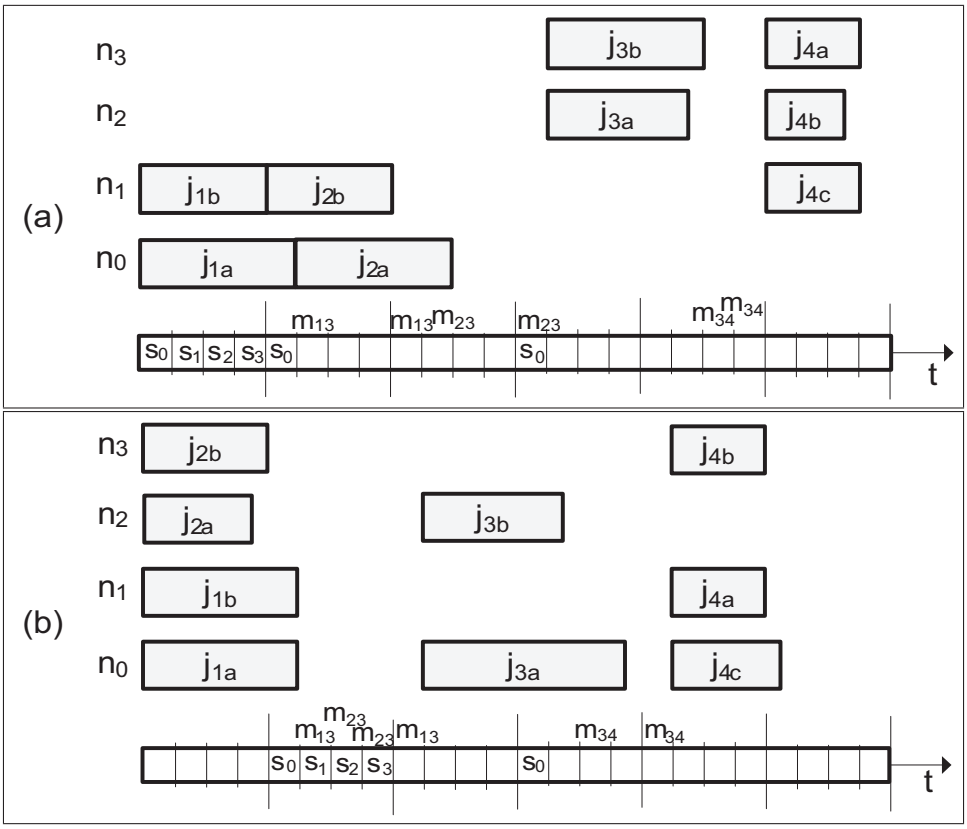

Figure 8.2: Assignment on a heterogeneous architecture

and $(16+20)\times 1.75 = 63$ respectively. According to *HetHeus* j_{3a} is assigned to node n_0, which has resulted the smallest product value. Similarly j_{3b} is assigned to n_2. In this way all other jobs are assigned. When there is a tie, a node is chosen arbitrarily. We observe that high reliable nodes (less failure probability) execute more jobs while maintaining the scheduling length to a minimum. There exists always a trade-off between reliability and scheduling length. In order to maintain a better trade-off between them, this heuristic can be applied stand alone or can also be applied when there is a tie in the communication heuristic described in Section 4.3.1.

Quality Mapping

The metrics in terms of influence, scheduling length and communication load have been calculated for both mappings. The values are $0.58, 92$ms, 150bytes and $0.43, 83$ms,150bytes which correspond to the assignment shown in Figure 8.2 (a) and (b) respectively. These objectives are used for measuring the

quality of the mapping. The overall influence and the scheduling length have been calculated by using the formulae given in Equation 5.6 and in Equation 5.9 respectively. The communication overhead is calculated by the sum of size of the messages transferred over the network. Both mappings satisfy all the constraints, i.e, both are feasible mapping. However mapping shown in Figure 8.2 (b) should be chosen as good mapping due to its less overall influence and scheduling length value. Note that this mapping has been created by applying the heuristics. As described earlier, these three variables have also been used for the optimization framework, where the experimental results show a significant quantitative gain.

8.2 Reliability Measure

Reliability is a property of dependability [3]. The reliability of a mapping is defined as the probability that all the assigned jobs are operational during a time period often called as mission time. In this section we present the estimation of reliability and some techniques of how to improve the reliability during the design process. Note that the measure of the reliability of a mapping becomes an important factor when nodes have different failure rates (*failure/hour*) and have different processor speeds. At the design time meaning that while conducting the mapping the reliability hence the dependability can be improved by using several techniques, such as:

(i) Replication of critical jobs,

(ii) Assigning job onto a node which has a less failure probability,

(iii) Assigning more complex job (incurs large CT) onto a high speed node processor,

(iv) Assign jobs onto nodes according to the minimum product of sum of execution time of the already allocated jobs and the corresponding node failure probability. This gives a better trade-off between reliability and schedulability, and

(v) Putting highly communicating jobs onto the same node decreases the system failure probability by avoiding the cascading failure and failure due to the fault in communication link, i.e., increases the system reliability.

In the following section we depict how we compute reliability and illustrate the improvement of reliability using an example.

8.2.1 Computing Reliability

The reliability $\hat{\Re}_l$ of a mapping is computed by the following expression:

$$\hat{\Re}_l = \prod_{i=1}^{n} \left(1 - (1 - r_{k_i})^{dc_i}\right) \tag{8.1}$$

where n is the number of jobs, r_{k_i} is the probability that a job/replica i operates correctly on k^{th} node and dc_i is the degree of replication for the critical jobs. The probability that a job/replica operates correctly depends on: *(i)* node processor n_k does not fail and *(ii)* incoming communication link is operational. Therefore, $r_{k_i} = r_{proc} \cdot r_{link}$. Where r_{proc} is the reliability factor depends on the failure rate of the processor and r_{link} is the reliability depends on the failure rate of the communication link.

$r_{proc} = e^{-\lambda_{nk} \cdot CT_i}$ is the probability that job j_i run successfully without the failure of its host processor. A job which takes more time to execute on a processor has a higher probability to fail than the ones which takes less time. Of course this is also depends on the failure probability (λ_{nk}) of the processor. The higher the failure rate of the processor is the higher the chance that a job running on that processor may fail. The reliability factor due to jobs communicating over the network is calculated as: $r_{link} = e^{-\lambda_l \cdot msg_{i,j}/b_n}$, where $\lambda_l \cdot msg_{i,j}/b_n$ is the unreliability factor due to inter-job communication between job j_i and j_j. This factor is counted only when two communicating jobs are assigned onto different nodes. It depends on the failure rate of the communication link λ_l as well as the time taken by transmitting the messages over the network. The larger is the message size the higher is the probability that the message get faulty/corrupted.

All the variables in the optimization framework are considered as minimization variable meaning that the lower the value of a variable is better for the design. Therefore in the optimization process the unreliability factor $\mu\Re_l$ is used while considering with other variables. The unreliability factor is equal to $1 - \hat{\Re}_l$.

8.2.2 Reliability and Mapping Analysis – An Example

We first take a very simple example to discuss the replication and reliability analysis when considering node processors of different failure rates. Let us assume an example which consists of two replicas j_{1a} and j_{1b}. We need to find a mapping to assign them onto two nodes from three nodes n_1, n_2 and n_3 in such a way that the reliability of the mapping is maximized. We assume that the job run on node n_1 and n_3 with higher reliability than on n_2. The reliability of a job run on nodes n_1, n_2 and n_3 is $r_{k_1} = 0.9, r_{k_2} = 0.8$ and

$r_{k3} = 0.9$ respectively. If we consider a mapping such as j_{1a} is assigned onto n_1 and j_{1b} is assigned onto n_2 then the reliability of the mapping is $\Re_l = 1 - (1 - .9)(1 - .8) = 0.98$. Now consider a different mapping where j_{1a} is assigned onto n_1 and j_{2b} is assigned onto n_3, in this case the overall reliability of the mapping becomes: $\Re_l = 1-(1-.9)(1-.9) = 0.99$. Obviously the second mapping configuration would be chosen as a better mapping with respect to the reliability. We observe that the reliability is dependent on the assignment policy when considering the node processors of having different failure probabilities. Furthermore in one hand replication of jobs increase the reliability and on the other hand the assignment policy of replicas impact the reliability.

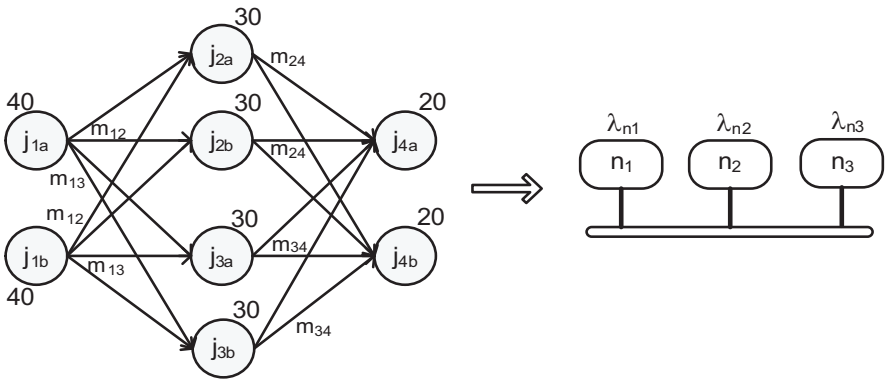

Figure 8.3: Application and architecture model

We now consider the example presented in Section 5.2.4 to be assigned onto an architecture of three nodes n_1, n_2 and n_3 with different failure rates of $\lambda_{n1} = 10^{-3}/h, \lambda_{n2} = 2 \times 10^{-4}/h$ and $\lambda_{n3} = 5 \times 10^{-4}/h$ respectively. All the jobs j_1, j_2, j_3 and j_4 are replicated two times and the assumed CTs are $40, 30, 30$ and 20ms respectively. The application (after replication) and the HW architecture is shown in Figure 8.3. All the 8 jobs (after replication the workload becomes double) have to finish their execution by 200ms. Assume that there is a probability of communication link failure at a rate of $10^{-3}/h$. With this given workload the reliability varies with the different mapping configurations. We have examined three different mapping configurations and estimated the reliability. Note that considering more number of jobs the difference of resulting reliability will be more prominent.

The mapping with the highest reliability will be chosen as a better mapping, in this case the configuration shown in Table 8.1 will be opted as a better mapping from a reliability view point. Note that when we use the HW architecture comprising the node processors of equal failure rates then

Node	Assigned jobs	Reliability $\Re_l$
n_1	j_{1a}, j_{2a}, j_{4a}	0.9999831
n_2	j_{1b}, j_{2b}, j_{3a}	
n_2	j_{3b}, j_{4b}	

Table 8.1: Mapping configuration (a) and reliability

Node	Assigned jobs	Reliability $\Re_l$
n_1	j_{3b}, j_{4b}	0.9999829
n_2	j_{1b}, j_{2b}, j_{3a}	
n_2	j_{1a}, j_{2a}, j_{4a}	

Table 8.2: Mapping configuration (b) and reliability

Node	Assigned jobs	Reliability $\Re_l$
n_1	j_{1b}, j_{2b}, j_{3a}	0.999974
n_2	j_{3b}, j_{4b}	
n_2	j_{1a}, j_{2a}, j_{4a}	

Table 8.3: Mapping configuration (c) and reliability

all the mapping configurations would return same reliability.

8.3 System Level Power Optimization

Given the increasing fuel prices in the world wide market, recently power/energy consumption has been attracted a significant research focus. Hence it is important to introduce a new dimension of power consumption into the system level co-design process. By the reduction of power consumption, for example in cars, cost and weight can be reduced as it may require low size generator and battery, for example, in car [102]. When the engine of a car is on all the networks and nodes are actively running almost all the functionalities, therefore the allocation cannot influence the power consumption much at this point. However dynamic power consumption minimization techniques can be applied to reduce the power at run time. Some functions/applications may require power until a certain time and some other functions require to provide services all the time when the car is stopped/parked. For these kind of applications saving power statically or at design time is important and this power consumption can be utilized by the allocation process. In the following we discuss methods and techniques of how to reduce the power consumption

for both static and dynamic power cases.

8.3.1 Static Power Optimization

The static or at design time power consumption minimization may depend on the mapping decision particularly when the engine is off and the ignition is off. As mentioned some functionalities need to run all the time and some need to run a certain time period. Hence, the power consumption is significantly important as the availability of services of functionalities depends on the battery life time. It is not necessarily the case that as big or powerful battery as we want is possible to install onboard a car. We are constrained by the size, weight and cost. For a comfort application like radio a limited time of availability though may not cause a problem but for safety relevant applications such as hazards light, parking light, it may not be acceptable to keep running the functions for a limited time period. There are several types of functions which need to be activated or to provide services on demand at any time, e.g., cost-critical applications like anti-theft protection, central locking system. For instance the sensors and actuators attached to those functionalities need to be activated all the time. Jobs from these type of applications can be grouped and assigned or allocated onto a common node then it would be sufficient to keep on only this node instead of spreading them over different nodes. The battery life time is increased by the consumption of less power. On the other hand when less amount of power is needed a small size of the battery would be sufficient. These types of assignment decisions can be taken into account in the ordering heuristics (allocation compatibility phase) of our mapping approach.

8.3.2 Dynamic Power Optimization

We use replication based FT scheme to tolerate both permanent and transient faults. More replicas consume relatively more power as more jobs need to be executed on processors. System level low power techniques should judiciously manage the replication resources to reduce the power consumption [179]. There are several techniques exists one could apply for the reduction of power consumption during the system design. The in-between slack (IBS) defined in Section 5.2.2, which is the gap between two consecutive jobs executing on the same node in a schedule, can be utilized for saving the power dynamically. In this section we simply term this as slack. The slack that cannot be used to run a complete job can be utilized for power saving or for future upgrading of a particular job. However slack may remain more than the amount of the computation time of a job which can be used for providing FT. In this section

we discuss how this slack can be utilized for power/energy saving. Obtain a snapshot view of remaining slacks and the amount of the slack from a schedule then use this information to set the new voltage or frequency level for a processor. However the slack can be utilized in an efficient way by applying the technique such as contingency scheduling [77]. If it is necessary to change the scheduling in order to obtain more slack for a better power saving it should be done in a way that the scheduling of one processor should not affect the scheduling of other processors. The technique for dynamic voltage and frequency scaling (DVFS) is described below.

Usually slack is utilized both for recovery from faults and for energy savings in a way that the deadline of the application is maintained [95; 96; 97]. In this case only transient faults are tolerated. The recovery time is usually included with WCET. However this may not be the case for applications with short deadlines which require to provide instant services despite of faults. In our case as we use active replication based FT scheme, the remaining slack from the scheduling can be fully utilized for power saving.

Dynamic Voltage and Frequency Scaling (DVFS)

Dynamic voltage and frequency scaling (DVFS) [180] is a technique to save power by dynamically scale the processor frequency and by controlling the system supply voltage. Higher processor frequency (when the processor runs at maximum frequency or at the highest voltage) increases die temperature and creates thermal stresses on the die which results more transient faults during the system operation. Therefore use of DVFS by scaling down the voltage and consequently saving the power is also beneficial from the dependability view point. On the other hand, lower processor voltage likely to lead to lower noise margin and more transient faults (induce by cosmic particle). Hence the voltage level has to be maintained or a recovery mechanism has to be provided for the faults that occurred due to lower voltage. Processor has to capable of adjusting the speed continually on a different level within minimum and maximum frequency/voltage range. From this discussion, it entails that while considering DVFS, care has to be taken to set and control the processor frequency by scaling down/up the operating voltage. The overhead or the cost (usually in μs range) due to switching the voltage level also need to be taken care.

The system level power model presented in [152] is described, where the power consumption $\hat{P}_o$ of a computing system is measured by the following expression. The model is utilized for power saving by DVFS technique.

$$\hat{P}_o \;=\; P_s + \hbar(P_{ind} + P_d) \;=\; P_s + \hbar(P_{ind} + C_{ef} f^m) \tag{8.2}$$

Where P_s is the sleep power, which includes the power to maintain basic circuits and to keep the clock running. It can be dislodged only by powering off the whole system. P_{ind} is the frequency independent active power, which consists of the components of memory and processor power. This part of the power consumption can be efficiently dislodged by putting systems to sleep state and is independent of system supply voltage and frequency. It is possible to minimize this part of the power by the mapping decision for some special applications mentioned in the previous section. The frequency dependent active power $P_d = C_{ef} f^m$ includes processor dynamic power and any power that depends on system processing speeds. $\hbar$ equals 0 if the system is in sleep state and equals 1 otherwise. The effective switching capacitance C_{ef} and the dynamic power exponent m (in general, larger than or equal to 2) are system dependent constants [95] and f is the processing frequency, i.e., the speed of the processor. While processor voltage as well as the frequency is changed the CT of each job is needed to be updated accordingly. The new CT' of each job running on different node processors is calculated by using this relation:

$$CT' = CT \cdot f_{max}/f$$

Assume that $f_{max} = 1$ then the relation becomes $CT' = CT/f$. Where, f is the frequency of the corresponding scaled down voltage of the processor. When the slack is known, the new frequency can be determined as:

$$f = CT/CT' = CT/(CT + IBS)$$

As we see from the Equation 8.2, the higher the frequency is the higher the power consumption (depending on the active power component P_d). Therefore jobs execute on a lower frequency will consume less power than execute on a higher speed. However too low frequency could consume more power as it takes too long time to finish the job execution. It is necessary to scale up/down the processor frequency and run it with a moderate/appropriate voltage while saving the power.

8.3.3 FT and Power Analysis – An Example

We analyze the power consumption with respect to the FT scheme and show how the consumption of power can be reduced for a mapping by utilizing the slack. We illustrate this by using an application and a HW architecture comprises of two nodes, which needs to tolerate 1-fault (shown in Figure 8.4). The application consists of four jobs j_1, j_2, j_3 and j_4 associated with the CTs of $20, 20, 30$ and 20ms respectively. For simplification, the overhead for fault detection, recovery and the voltage switching time are included with

the corresponding job CT. All jobs have to finish their execution by the deadline of 160ms and each slot length is equal to 10ms. According to the FT requirement, the degree of criticality dc_i has been set at the application level equal to 2. Therefore all jobs under this application have been replicated two times and the corresponding messages have also been replicated. Figure 8.4 shows the application graph after replication. This FT scheme can tolerate either one permanent (fail-silent) or one transient fault. Assume that all the jobs run on each processor with maximum frequency of f_1 (the jobs with the higher width as shown in the figure).

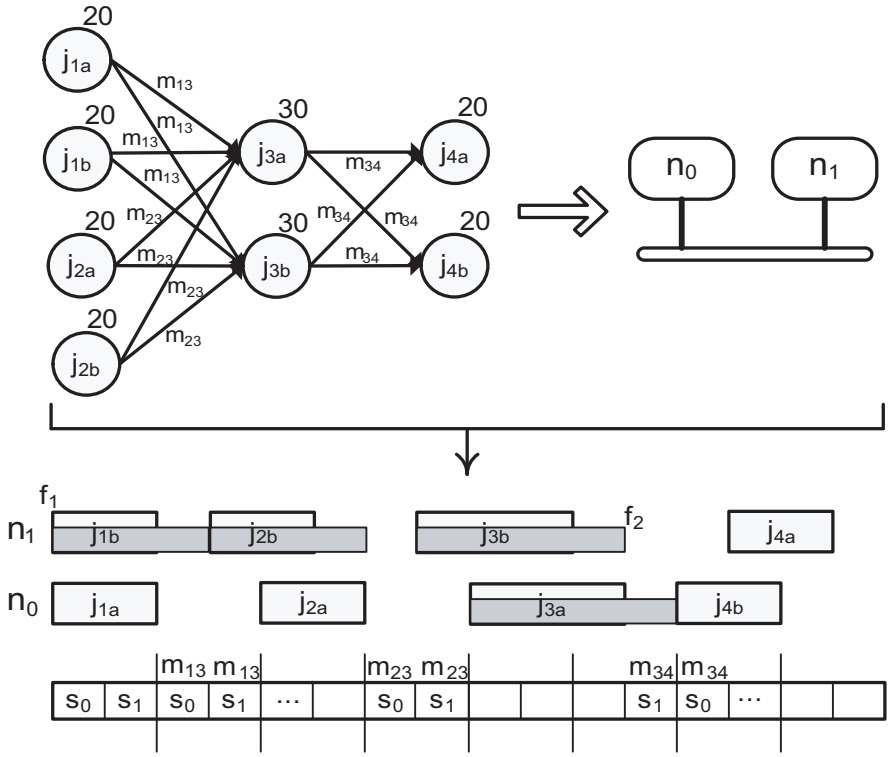

Figure 8.4: Example mapping - FT and power analysis

Reliability is increased by replicating more jobs which increases the number of jobs executing on different computing nodes resulting increased scheduling length and more power consumption. However techniques can be applied first to provide FT and then to minimize the power consumption while still satisfying the schedulability. Particularly the available slack can be utilized for power saving without impacting the existing scheduling. For example in Figure 8.4, job $j_{1b}, j_{2b}, j_{3b}, j_{3a}$ can run with long CT utilizing their front slack which will not impact the overall scheduling on any nodes (so called contingency scheduling). This can be done by scaling down the voltage that

run the processor on a lower frequency f_2 ($f_2 < f_1$). For instance, j_{1b} runs in a frequency of $f_2 = f_1 * 20/(20 + 10) = 0.67 * f_1$. As we see from the Equation 8.2 that if the frequency is low then less amount of power is required as well. Hence the overall power consumptions due to more replicas can be minimized. We calculate the energy consumption of the mapping shown in the figure for both cases (before and after DVFS). We assume that the processor of each node consume 10mW of power (the value of the power has been chosen according to AMD K6-2E 400MHz processor [181]). In the calculation, we use only the active power part $P_d = C_{ef} f^m$ as this the only part which depends on the frequency, where $m = 2.6$ [99] and C_{ef} is constant for both cases. Before applying the energy saving technique, the total energy consumption is 3000μJ and after DVFS technique it becomes 2126.33μJ. Therefore a clear reduction of energy consumption can be noticed for this small mapping configuration.

Chapter 9

Conclusions and Future Issues

In this thesis we have developed methods and techniques for designing integrated dependable real-time (RT) embedded systems. Different design concepts and strategies have been developed for the integration of varied criticality SW-Cs onto a common distributed HW architecture such that dependability/fault-tolerance (FT) and RT requirements are not compromised. Moving from the traditional federated design or customized design approach we provide design and optimization methodology for an integrated design architecture. The design methodology has been developed at system level, i.e., at higher abstraction level.

In the following we summarize the main contributions made in this thesis. Furthermore the chapter provides future research directions relevant to the thesis work.

9.1 The Overall Contributions

The goal of the thesis was to develop a dependability driven system level co-design and optimization methodology for embedded systems. This section summarizes the overall contributions.

9.1.1 The Integrated Design and Optimization Framework

We have developed a novel extra-functionality (FT, RT, resources and power) driven system level co-design methodology for designing FT-RT embedded systems and have presented in this thesis. To the best of our knowledge this is the first co-design/integration approach dealing at system level where the prime focus is on design and optimization of both dependability and

RT. The proposed approach meticulously guides the defined co-design steps presented in Chapter 3. The design starts from requirements analysis through mapping to its optimization and prototyping. Under this framework various embedded system co-design criteria and guidance for exploring and exploiting the design space have been presented. During the overall design process, we first have developed a heuristic based systematic mapping approach which produces an initial feasible solution. A mapping is feasible when it satisfies all the defined constraints. The optimization has then been performed from that feasible design point. Starting the optimization from a feasible point and maintaining the feasibility during the process reduces the search space considerably. Overall we have developed the design and optimization concept for an integrated embedded architecture.

Resultant Publications

- **Shariful Islam**, Neeraj Suri, András Balogh, György Csertán & András Pataricza, *A Transformation Based Design for Integrated Dependable Real-Time Embedded Systems*, Submitted to the Journal of Design Automation for Embedded Systems (DAES) (In review), 2008.

- **Shariful Islam** & Neeraj Suri, *A Multi Variable Optimization Approach for the Design of Integrated Dependable Real-Time Embedded Systems*, In the IFIP International Conference on Embedded and Ubiquitous Computing (EUC), 2007.

- **Shariful Islam**, Robert Lindström & Neeraj Suri, *Dependability Driven Integration of Mixed Criticality SW Components*, In the 9th IEEE International Symposium on Object and Component-oriented Real-time distributed Computing (ISORC), 2006.

9.1.2 Consolidated Mapping of Mixed Criticality Applications

A systematic and consolidated mapping of different criticality applications (both safety-critical and non-safety critical) onto a common distributed computing platform has been performed, thus designing an integrated system. The functional and extra-functional requirements have been considered at an early design stage, which reduces the design efforts. In the course of the system level co-design, we have presented comprehensive mapping strategies for ensuring FT (replicated jobs have been placed onto distinct nodes), enhancing dependability (reducing the influence, i.e., by reducing the error

propagation probability between nodes) and providing schedulability analysis (satisfying timeliness properties). These design strategies have been maintained during the optimization process as well. A so called job and node ordering heuristic has been used in the mapping algorithm to create the initial feasible mapping. By using these heuristics the mapping has been generated with least number of iterations. We have employed different constraints satisfaction techniques as well as the technique to increase the performance of creating the mapping such as constraints prioritization, consistency enforcing and backtracking.

Experimental results show the effectiveness, performance and robustness of our initial mapping process as the mapping considers the following important aspects: *(i)* uses of job and node ordering heuristics, *(ii)* checking the constraints during the assignment process (applying retrospective technique) and are satisfied as a basis of constraints prioritization which reduces the search space by avoiding unnecessary exploration of infeasible design space, *(iii)* as the algorithm uses the ordering heuristics and run into multiple phases it takes less number of iterations as well as less convergence time, and *(iv)* the generated initial solution is used to improve the performance of the optimization.

Resultant Publications

- **Shariful Islam**, Robert Lindström & Neeraj Suri, *Dependability Driven Integration of Mixed Criticality SW Components*, In the 9th IEEE International Symposium on Object and Component-oriented Real-time distributed Computing (ISORC), 2006.

- **Shariful Islam**, Neeraj Suri, András Balogh, György Csertán & András Pataricza, *A Transformation Based Design for Integrated Dependable Real-Time Embedded Systems*, Submitted to the Journal of Design Automation for Embedded Systems (DAES) (In review), 2008.

9.1.3 Extra-Functionality Driven Optimization

We have presented a generic Multi Variable Optimization (MVO) framework for the design of an integrated FT-RT embedded system. The framework depicts a composite extra-functionalities driven system design and optimization of various competing design variables from dependability, RT, resource and power perspectives. The optimization is an iterative improvement process. Multiple constraints and several variables have been considered simultaneously during the optimization process. A feasible mapping has been used

as an input to the MVO algorithm. However the feasibility of the mapping has been maintained by implementing an external function which check the constraints in each iteration. In this way the search process has always been directed in the feasible region of the global design space. The approach has been applied to an existing metaheuristic based optimization algorithm. For this purpose we have used simulated annealing which has converged to an optimized solution (a near-optimal solution is sufficed).

The experimental results show the effectiveness of the approach and a significant improvement of the proposed design compared to a straightforward solution where optimization has not been applied. We have shown how the variables are quantified and optimized. The variables include *influence, scheduling length* and *bandwidth utilization*. An MVO function has been constructed as weighted sum of the function of each variable which returns the overall value of a corresponding mapping (measure the quality of the mapping). We have discussed in Chapter 8 how the generic framework can be extended with additional variables, e.g., we provide quantification and optimization technique for reliability and power.

Quantification of Variables

The *quantification* and modeling of a set of variables for the design of mixed critical RT embedded systems have been given in detail. This includes how to estimate/measure variables, and how to formulate them in terms of *function minimization*. The quantification of variables is a prerequisite for performing the optimization. In this thesis we have quantified a set of variables from dependability/FT, RT and resource perspectives. In order to reduce the error propagation probability between different modules (either between jobs or between nodes), i.e., to confine the fault/error within a single node, we use the variable *influence*. For schedulability analysis and to reduce the length of the scheduling we have used *scheduling length*. In order to reduce the overhead on the network, *bandwidth utilization* technique has been employed. In addition we have provided the measure of reliability and power consumption.

Resultant Publication

- **Shariful Islam** & Neeraj Suri, *A Multi Variable Optimization Approach for the Design of Integrated Dependable Real-Time Embedded Systems*, In the IFIP International Conference on Embedded and Ubiquitous Computing (EUC), 2007.

Prototyping

We have designed a prototype for the developed process which integrates different tools for modeling the application to the deployment of a complete system. The post integration platform specific model generated by the tool-chain controls the deployment task of the system development process to run executables on the target platform. The implementation of custom user interface enhances the usability of the tool-chain providing convenient user interface, ease in importing models, one-click transformation, performing mapping algorithm, generating configuration files and results.

Resultant Publications

- **Shariful Islam**, György Csertán, András Balogh, Wolfgang Herzner, Thierry Le Sergent, András Pataricza & Neeraj Suri, *A SW-HW Integration Process for the Generation of Platform Specific Models*, Microelectronics (ME), 2006.

- Wolfgang Herzner, Martin Schlager, Thierry Le Sergent, Bernhard Huber, **Shariful Islam**, Neeraj Suri & András Balogh, *From Model-Based Design to Deployment of Integrated, Embedded, Real-Time Systems: The DECOS Tool-Chain*, Microelectronics (ME), 2006.

- Wolfgang Herzner, Rupert Schlick, Martin Schlager, Bernhard Leiner, Bernhard Huber, András Balogh, György Csertán, Alain LeGuennec, Thierry LeSergent, Neeraj Suri & **Shariful Islam**, *Model-Based Development of Distributed Embedded Real-Time Systems with the DECOS Tool-Chain*, In Proceedings of Society for Automotive Engineers (SAE) Aerotech, 2007.

9.1.4 Summarizing the Benefits

We emphasize the following preeminent benefits of our overall approach: *(i)* the developed methodology meticulously guides the design of integrated dependable RT embedded systems, *(ii)* dependability/FT is provided and is enhanced by restricting the possible nodes from correlated faults, *(iii)* RT requirements are met and the scheduling length is minimized, which increases the overall system performance, *(iv)* bandwidth utilization is reduced, which allows the use of a slower but cheaper bus, *(v)* the optimization enables the usage of less number of resources/nodes, integration of new functionality and the future upgrading of possible functions, *(vi)* extendability and adaptability of the design approach, and *(vii)* we believe that the developed co-design

methodology and prototype will reduce the design and development efforts and time.

9.2 Future Issues

The developed methods and techniques presented in this thesis have opened up new working directions. We discuss some issues as follows. We have provided the concept of integration of varied criticality applications onto a common distributed HW architecture. However it would be interesting to apply the concept on a MPSoC (Multiprocessor System on Chip) architecture. Moreover an online adaptation (specially for non-SC applications) of the approach or a reconfiguration technique can be applied. We also list the following issues as part of the future work.

- We have provided an integration/co-design process considering the architecture of symmetric multiprocessing (SMP) and have briefly discussed the applicability on a heterogeneous architecture comprises processors of different speeds and failure rates. In future this issue of heterogeneity needs to be elaborated.

- Both transient and permanent faults are assumed in the current fault model, therefore we apply active SW based replication as an FT scheme. If only transient faults are assumed to be tolerated then replication, re-execution, checkpointing or interplay of these techniques can be applied. A further investigation is necessary for an optimal FT technique (when only transient faults are assumed) such that RT properties are maintained and resources are properly utilized.

- The developed system level co-design is a generic framework which we have showed by extending it adding more design variables during the optimization phase. The optimization process can be performed adding more variables (reliability and power consumption). Currently, we have provided the quantification of these variables. However they need to be used in the optimization algorithm.

- To find an optimized solution the MVO approach has been applied to an existing algorithm called simulated annealing. However as mentioned other techniques such as tabu search or genetic algorithm can also be applied.

Integration on MPSoC Architecture and Reconfiguration: System-on-Chip (SoC) is a complete digital system build on a single chip and often contains multiple embedded processors leading to an Embedded SoC (ESoC) or MPSoC design. MPSoC contains heterogeneous processors for better performance and power achievement [182]–p 279. We have presented the applicability of the approach on a heterogeneous architecture which might also be useful for the MPSoC architecture. The challenges remain on the integration/mapping of mixed criticality applications onto ESoC or MPSoC architecture while still providing FT, fault isolation and encapsulation, i.e., maintaining the separation between different criticality applications. Such an architecture has been developed in [183], where multiprocessor SoC with a time-triggered (TT) network-on-chip formed an architecture called as TT-SoC. A job introduced in this thesis can be run on a microcomputer (self-computing element) of a TT-SoC architecture. A further elaboration on the architecture providing FT and fault isolation is developed in [184]. [185] describes the mapping and optimization approaches on a MPSoC while [186] discusses about considering the dependability properties. The method and technique for developing the heuristic based mapping and optimization presented in this thesis would be a viable strategy for developing the design concept for MPSoC architecture.

The developed methods and techniques are applied statically at design time to provide predictable and deterministic behavior of safety-critical systems. However the approach can be adapted to run time design. The design can be reconfigured depending on the run time behavior of the system. When there is a fault in a node or in a communication link, the affected jobs on that specific node need to be re-executed on different nodes (a different mapping configuration), in order to achieve a certain level of dependability/FT. This can be performed by triggering the execution of a faulty job on another node. If there is a fault in the communication link then jobs which uses that link need to be re-executed on different nodes such that they can communicate via different link. A set of different mapping configurations can be created *a priori* and an appropriate mapping can be selected at run time according to the need of the overall system dependability requirements. An online approach for SW-HW co-design (online SW-HW partitioning) is presented in [187], where the author develops a fault tolerant and self-adaptive technique for reconfigurable networked embedded systems. The approach is applied at system level and allows functionality move from one node to another at run time.

Bibliography

[1] B. Bouyssounouse & J. Sifakis, Current Design Practice and Needs in Selected Industrial Sectors, *Artist FP5 Consortium: Embedded Systems Design*, LNCS 3436, pages 15 - 38, 2005.

[2] J. Dannenberg & C. Kleinhans, The coming Age of Collaboration in the Automotive Industry, *Mercer Management Journal*, 17, pages 88-94, 2004.

[3] J-C., Laprie, & B. Randell, Basic Concepts and Taxonomy of Dependable and Secure Computing, *IEEE Transactions on Dependable Secure Computing (TDSC)*, 1(1), pages 11–33, 2004.

[4] V. Claesson, S. Poledna & J. Soderberg, The XBW Model for Dependable Real-Time Systems, *International Conference on Parallel and Distributed Systems (ICPADS)*, pages 130–138, 1998.

[5] C. Wilwert, N. Navet, Y.-Q. Song & F. Simonot-Lion, Design of Automotive X-by-Wire Systems, *In The Industrial Communication Technology Handbook, CRC Press*, 2004.

[6] X-by-Wire Project, Brite-EuRam 111 Program, X-By-Wire - Safety Related Fault Tolerant Systems in Vehicles, *Final Report*, 1998.

[7] S. Kumar, J. H. Aylor, B. W. Johnson & W. A. Wulf, The Codesign of Embedded Systems: A Unified Hardware/Software Representation, *Kluwer Academic Publishers*, 1996.

[8] B. Tabbara, A. Tabbara & A. Sangiovanni-Vincentelli, Function/Architecture Optimization and Co-Design of Embedded Systems, *Kluwer Academic Pblishers*, 2000.

[9] J. Staunstrup & W. Wolf (eds.), Hardware/Software Co-Design: Principles and Practice, *Kluwer Academic Publishers*, 1997.

[10] M. Broy, I.H. Kruger, A. Pretschner & C. Salzmann, Engineering Automotive Software, *Proceedings of the IEEE*, 95(2), pages 356–373, 2007.

[11] N. Navet, Y. Song, F. Simonot-Lion and C. Wilwert, Trends in Automotive Communication Systems, *Proceedings of the IEEE*, 93(6), pages 1204–1223, 2005.

[12] H. Kopetz, R. Obermaisser, P. Peti & N. Suri, From a Federated to an Integrated Architecture for Dependable Embedded Real-Time Systems. *Technical Report 22, Institut für Technische Informatik, Technische Universität Wien*, Treitlstr. 1-3/182-1, 2004.

[13] CAN Specification, Controller Area Network Specification and Implementation, *Robert Bosch GmbH*, http://www.semiconductors.bosch.de/pdf/can2spec.pdf, 1991.

[14] LIN Specification, Local Interconnect Network, http://www.linsubbus.org/, Accessed 21 July, 2008.

[15] MOST Specification, Media Oriented Systems Transport, Rev 3.0, 05/2008, http://www.mostcooperation.com/, Accesses 21 July, 2008.

[16] H. Kopetz & G. Grünsteidl, TTP - A Protocol for Fault-Tolerant Real-Time Systems, *Journal of Computer*, 27(1), pages 14–23, 1994.

[17] The FlexRay Group, FlexRay Communications System Protocol Specification, Version 2.1, http://www.flexray.com/, 2005.

[18] H. Kopetz, A. Ademaj, P. Grillinger & K. Steinhammer, The Time-Triggered Ethernet (TTE) Design, *IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 22–33, 2005.

[19] ARINC, Design Guidance for Integrated Modular Avionics, *Aeronautical Radio Inc., ARINC Report 651*, 1991.

[20] Y.-H. Lee, D. Kim, M. Younis, J. Zhou & J. McElroy, Resource Scheduling in Dependable Integrated Modular Avionics, *International Conference on Dependable Systems and Networks (DSN)*, pages 14–23, 2000.

[21] M. F. Younis, M. Aboutabl & D. Kim, Software Environment for Integrating Critical Real-Time Control Systems, *Journal of System Architecture*, 50(11), pages 649–674, 2004.

[22] AUTOSAR, Technical Overview V2.0.1, AUTOSAR GbR 2006.

[23] P. Peti, R. Obermaisser, F. Tagliabo, A. Marino & S. Cerchio, An Integrated Architecture for Future Car Generations, *IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 2–13, 2005.

[24] H. Kopetz & G. Bauer, The Time-Triggered Architecture, *Proceeding of the IEEE*, 91(1), pages 112–126, 2003.

[25] M. Broy, Automotive Software and Systems Engineering, *ACM and IEEE International Conference on Formal Methods and Models for Co-Design (MEMCOD)*, pages 143–149, 2005.

[26] P. Pop, P. Eles, Z. Peng & T. Pop, Analysis and Optimization of Distributed Real-Time Embedded Systems, *ACM Transactions on Design Automation Electronic Systems*, 11(3), pages 593–625, 2006.

[27] A. Berger, Embedded Systems Design: An Introduction to Processes, Tools and Techniques, *CMP Books, USA*, 2002.

[28] E. Dubrova, Fault-Tolerant Design: An Introduction, *Kluwer Academic Publishers*, 2007.

[29] A. Johansson, Robustness Evaluation of Operating Systems, *PhD Thesis, Technische Universität Darmstadt*, 2008.

[30] N. Suri, M. Hugue & C.J. Walter, Advances in Ultra-Dependable Distributed Systems, *IEEE Computer Society Press*, 1995.

[31] B. Bouyssounouse & J. Sifakis, Embedded Systems Design: The ARTIST Roadmap for Research and Development, *Springer-Verlag*, 2005.

[32] J. Rushby, Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance, *SRI International, NASA/CR-1999-209347*, 1999.

[33] A. Jhumka, S. Klaus & S. A. Huss, A Dependability-Driven System-Level Design Approach for Embedded Systems, *Design, Automation and Test in Europe Conference and Exposition (DATE)*, pages 372–377, 2005.

[34] D. Fernandez-Baca, Allocating Modules to Processors in a Distributed System, *IEEE Transactions on Software Engineering*, 15(11), pages 1427–1436, 1989.

[35] M. R. Garey & D. S. Johnson, Computers and Intractability : A Guide to the Theory of NP-Completeness, *W. H. Freeman*, 1979.

[36] A. Damm, J. Reisinger, W. Schwabl & H. Kopetz, The Real-Time Operating System of MARS, *ACM SIGOPS Operating Systems Review*, 23(3), pages 141–157, 1989.

[37] S. Wang, J. R. Merrick & K. G. Shin, Component Allocation with Multiple Resource Constraints for Large Embedded Real-Time Software Design, *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 219–226, 2004.

[38] S. Ghosh, R. Rajkumar, J. Hansen & J. Lehoczky, Scalable Resource Allocation for Multi-Processor QoS Optimization, *International Conference on Distributed Computing Systems (ICDCS)*, pages 174-183, 2003.

[39] M. Eisenring, L. Thiele & E. Zitzler, Conflicting Criteria in Embedded System Design, *IEEE Design & Test of Computers*, 17(2), pages 51–59, 2000.

[40] A. Sangiovanni-Vincentelli & G. Martin, Platform-Based Design and Software Design Methodology for Embedded Systems, *IEEE Design & Test of Computers*, 18(6), pages 23–33, 2001.

[41] A. Sangiovanni-Vincentelli & M. Di Natale, Embedded System Design for Automotive Applications, *Journal of Computer*, 40(10), pages 42–51, 2007.

[42] A. Balogh, A. Pataricza & J. Rácz, Scheduling of Embedded Time-Triggered Systems, *Proceedings of the Workshop on Engineering Fault Tolerant Systems (EFTS)*, pages 44–49, 2007.

[43] H. Kopetz, Real-Time Systems, Design Principles for Distributed Embedded Applications, *Kluwer Academic Publishers*, 1997.

[44] J. A. Stankovic & K. Ramamritham, Tutorial: Hard Real-Time Systems, *IEEE Computer Society Press*, 1989.

[45] R. Kirner & P. Puschner, Classification of WCET Analysis Techniques, *IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 190–199, 2005.

[46] J. Staschulat, J. C. Braam, R. Ernst, T. Rambow, R. Schlor & R. Busch, Cost-Efficient Worst-Case Execution Time Analysis in Industrial Practice, *International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*, pages 204–211, 2006.

[47] International Electrotechnical Commission, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (IEC 61508), International Electrotechnical Commission, http://www.iec.ch/, Accessed 22 August 2008.

[48] J. L. Silva, Metaheuristic and Multiobjective Approaches for Space Allocation, *University of Nottingham, PhD thesis*, 2003.

[49] O. Rossi-Doria, & B. Paechter, An Hyperheuristic Approach to Course Timetabling Problem Using an Evolutionary Algorithm, *Technical Report CC-00970503, Napier University, Scotland*, 2003.

[50] N. Sadeh & M.S. Fox, Variable and Value Ordering Heuristics for the Job Shop Scheduling Constraint Satisfaction Problem, *Artificial Intelligence*, 86(1), pages 1–41, 1996.

[51] S. Kirkpatrick, C. D. Gelatt & M. P. Vecchi, Optimization by Simulated Annealing, *Journal of Science*, 220(4598), pages 671–680, 1983.

[52] C. Blum & A. Roli, Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison, *ACM Computing Surveys (CSUR)*, 35(3), pages 268–308, 2003.

[53] R. Bellman, Dynamic Programming, *Dover Publications, Incorporated*, 2003.

[54] M. Lundy & A Mees, Convergence of an Annealing Algorithm, *Mathematical Programming: Series A and B*, 34(1), pages 111–124, 1986.

[55] F. Glover, Tabu Search – Part I, *Journal on Computing*, 1(3), pages 190–206, 1989.

[56] D. A. Coley, An Introduction to Genetic Algorithms for Scientists and Engineers, *World Scientific Publishing*, 1999.

[57] W. Wolf, A Decade of Hardware/Software Codesign, *Journal of Computer*, 3(4), pages 38–43, 2003.

[58] J. Madsen, J. Grode, P. V. Knudsen, M. E. Petersen & A. Haxthausen, LYCOS: The Lyngby Co-Synthesis System, *Journal of Design Automation for Embedded System*, 2(2), pages 195–235, 1997.

[59] J. Henkel & R. Ernst, An Approach to Automated Hardware/Software Partitioning using a Flexible Granularity that is Driven by High-level Estimation Techniques, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(2), pages 273–289, 2001.

[60] F. Balarin, E. Sentovich, M. Chiodo, P. Giusto, H. Hsieh, B. Tabbara, A. Jurecska, L. Lavagno, C. Passerone, K. Suzuki & A. Sangiovanni-Vincentelli, Hardware-Software Co-Design of Embedded Systems: The POLIS Approach, *Kluwer Academic Publishers*, 1997.

[61] T. Blickle, J. Teich & L. Thiele, System-Level Synthesis using Evolutionary Algorithms, *Journal of Design Automation for Embedded Systems*, 3, pages 23–58, 1998.

[62] A. D. Pimentel, C. Erbas & S. Polstra, A Systematic Approach to Exploring Embedded System Architectures at Multiple Abstraction Levels, *IEEE Transactions on Computers*, 55(2), pages 99–112, 2006.

[63] A. Sangiovanni-Vincentelli, Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design, *Proceedings of the IEEE*, 95(3), pages 467–506, 2007.

[64] C. Bolchini, L. Pomante, F. Salice & D. Sciuto, Reliability Properties Assessment at System Level: A Co-Design Framework, *Journal of Electronic Testing: Theory and Applications*, 18(3), pages 351–356, 2002.

[65] J. Axelsson, HW/SW Codesign for Automotive Applications: Challenges on the Architecture Level, *IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 123–126, 2001.

[66] R. von Hanxleden, A. Botorabi & S. Kupczyk, A Codesign Approach for Safety-Critical Automotive Applications, *IEEE Micro*, 18(5), pages 66–79, 1998.

[67] C. Ekelin & J. Jonsson, Evaluation of Search Heuristics for Embedded System Scheduling Problems, *Principles and Practice of Constraint Programming (CP)*, pages 640–654, 2001.

[68] K. Kuchcinski, Constraints-Driven Scheduling and Resource Assignment, *ACM Transaction on Design Automation of Electronic Systems (TODAES)*, 8(3), pages 355–383, 2003.

[69] C.-J. Hou & K. G. Shin, Replication and Allocation of Task Modules in Distributed Real-Time Systems, *International Symposium on Fault-Tolerant Computing (FTCS-24)*, pages 26–35, 1994.

[70] D. Peng, K. G. Shin & T. F. Abdelzaher, Assignment and Scheduling of Communicating Periodic Tasks in Distributed Real-Time Systems, *IEEE Transactions on Software Engineering*, 23(12), pages 745–758, 1997.

[71] T. F. Abdelzaher & K. G. Shin, Combined Task and Message Scheduling in Distributed Real-Time Systems, *IEEE Transactions on Parallel and Distributed Systems (IPDS)*, 10(11), pages 1179-1191, 1999.

[72] R. Rajkumar, C. Lee, J. P. Lehoczky & D. P. Siewiorek, Practical Solutions for QoS-Based Resource Allocation, *IEEE Real-Time Systems Symposium (RTSS)*, pages 296–306, 1998.

[73] S. Kodase, S. Wang, Z. Gu & K. Shin, Improving Scalability of Task Allocation and Scheduling in Large Distributed Real-Time Systems using Shared Buffers, *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 181–188, 2003.

[74] C. L. Liu & J. W. Layland, Scheduling Algorithms for Multi-Programming in a Hard Real-Time Environment, *Journal of the Association for Computing Machinery*, 20(1), pages 40–61, 1973.

[75] Y. Oh, & S. H. Son, Enhancing Fault-Tolerance in Rate-Monotonic Scheduling, *Real-Time System: Special Issue on Responsive Computer Systems*, 7(3), pages 315–329, 1994.

[76] S. Ghosh, R. Melhem & D. Mossé, Enhancing Real-Time Schedules to Tolerate Transient Faults, *IEEE Real-Time Systems Symposium (RTSS)*, pages 120–129, 1995.

[77] N. Kandasamy, J. P. Hayes & B. T. Murray, Tolerating Transient Faults in Statically Scheduled Safety-Critical Embedded Systems, *IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 212–221, 1999.

[78] J. Yuan, C. Pixley & A. Aziz, Constraint-Based Verification, *Springer Science+Business Media, Inc., New York, USA*, 2006.

[79] N. Suri, S. Ghosh & T. Marlowe, A Framework for Dependability Driven Software Integration, *International Conference on Distributed Computing Systems (ICDCS)*, pages 406–415, 1998.

[80] S. Mustafiz & J. Kienzle, A Survey of Software Development Approaches Addressing Dependability, *Scientific Engineering of Distributed Java Applications(FIDJI)*, pages 78–90, 2004.

[81] V. M. Lo, Heuristic Algorithms for Task Assignment in Distributed Systems, *IEEE Transaction on Computer*, 37(11), pages 1384–1397, 1988.

[82] C.-J. Hou, & K.G. Shin, Allocation of Periodic Task Modules with Precedence and Deadline Constraints in Distributed Real-Time Systems, *IEEE Transaction on Computer*, 46(12), pages 1338–1356, 1997.

[83] M. D. Natale & J.A. Stankovic, Scheduling Distributed Real-Time Tasks with Minimum Jitter, *IEEE Transaction on Computer*, 49(4), pages 303–316, 2000.

[84] K. Tindell, A. Burns & A. Wellings, Allocating Hard Real-Time Tasks: An NP-Hard Problem Made Easy, *Real-Time System*, 4(2), pages 145–165, 1992.

[85] V. Izosimov, P. Pop, P. Eles & Z. Peng, Design Optimization of Time- and Cost-Constrained Fault-Tolerant Distributed Embedded Systems, *Design, Automation and Test in Europe Conference and Exposition (DATE)*, pages 864–869, 2005.

[86] S. M. Shatz, J. Wang & M. Goto, Task Allocation for Maximizing Reliability of Distributed Computer Systems, *IEEE Transaction on Computer*, 41(9), pages 1156–1168, 1992.

[87] S. Kartik & C. Murthy, Task Allocation Algorithms for Maximizing Reliability of Distributed Computing Systems, *IEEE Transaction on Computer*, 46(6), pages 719–724, 1997.

[88] P.-Y. Yin, S.-S. Yu, P.-P. Wang & Y-T Wang, Task Allocation for Maximizing Reliability of a Distributed System using Hybrid Particle Swarm Optimization, *Journal of Systems and Software*, 80(5), pages 724–735, 2007.

[89] F. Bicking, B. Conrard & J.-M. Thiriet, Integration of Dependability in a Task Allocation Problem, *IEEE Transactions on Instrumentation and Measurement*, 53(6), pages 1455–1463, 2004.

[90] I. Bate & P. Emberson, Design for Flexible and Scalable Avionics Systems, *IEEE Aerospace Conference*, pages 1–12, 2005.

[91] I. Assayad, A. Girault & H. Kalla, A Bi-Criteria Scheduling Heuristic for Distributed Embedded Systems under Reliability and Real-Time Constraints, *International Conference on Dependable Systems and Networks (DSN)*, pages 347–356, 2004.

[92] A. Dogan, & F. Özgüner, Biobjective Scheduling Algorithms for Execution Time-Reliability Trade-off in Heterogeneous Computing Systems, *Journal of Computer*, 48(3), pages 300–314, 2005.

[93] W. Luo, X. Qin & K. Bellam, Reliability-Driven Scheduling of Periodic Tasks in Heterogeneous Real-Time Systems, *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW)*, pages 778–783, 2007.

[94] D. Zhu & H. Aydin, Energy Management for Real-Time Embedded Systems with Reliability Requirements, *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 528–534, 2006.

[95] D. Zhu & H. Aydin, Reliability-Aware Energy Management for Periodic Real-Time Tasks, *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 225–235, 2007.

[96] R. Melhem, D. Mosse & E. Elnozahy, The Interplay of Power Management and Fault Recovery in Real-Time Systems, *IEEE Transaction on Computer*, 53(2), pages 217–231, 2004.

[97] Y. Zhang & K. Chakrabarty, A Unified Approach for Fault Tolerance and Dynamic Power Management in Fixed-Priority Real-Time Embedded Systems, *IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems*, 25(1), pages 111–125, 2006.

[98] G. Palermo, C. Silvano & V. Zaccaria, Multi-Objective Design Space Exploration of Embedded Systems, *Journal of Embedded Computing*, 1(3), pages 305–316, 2005.

[99] D. Zhu, R. Melhem, D. Mosse & E. Elnozahy, Analysis of an Energy Efficient Optimistic TMR Scheme, *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS)*, pages 559–568, 2004.

[100] U. Bordoloi & S. Chakraborty, Performance Debugging of Real-Time Systems Using Multicriteria Schedulability Analysis, *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 193–202, 2007.

[101] C. Ekelin, An Optimization Framework for Scheduling of Embedded Real-Time Systems, *PhD Thesis, Chalmers University of Technology*, 2004.

[102] B. Hardung, Optimisation of the Allocation of Functions in Vehicle Networks, *PhD Thesis, Universität Erlnagen-Nürnberg*, 2006.

[103] P. Leteinturier, Multi-Core Processors: Driving the Evolution of Automotive Electronics Architectures, *Published on Embedded.com dated 16/09/07*, http://www.embedded.com/design/multicore/201806714, Accessed 23 July, 2008.

[104] J. Rushby, Bus Architectures for Safety-Critical Embedded Systems, *In T. Henzinger and C. Kirsch, editors, Embedded Software (EMSOFT)*, LNCS 2211, pages 306–323, 2001.

[105] R. Obermaisser, P. Peti & H. Kopetz, Virtual Networks in an Integrated Time-Triggered Architecture, *IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS)*, pages 241–253, 2005.

[106] R. Seyer, C. Siemers, R. Falsett, K. Ecker & H. Richter, Robust Partitioning for Reliable Real-Time Systems, *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, pages 117–122, 2004.

[107] P. Parkinson & L Kinnan, Safety-Critical Software Development for Integrated Modular Avionics, *White Paper, Wind River Systems, Inc.*, 2006.

[108] H. Kopetz & N. Suri, Compositional Design of RT Systems: A Conceptual Basis for Specification of Linking Interfaces, *IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 51–60, 2003.

[109] N. Kandasamy, J. P. Hayes & B. T. Murray, Dependable Communication Synthesis for Distributed Embedded Systems, *Computer Safety, Reliability and Security Conference (SAFECOM)*, LNCS 2788, pages 275–288, 2003.

[110] J. Liu, P. H. Chou, N. Bagherzadeh & F. Kurdahi, A Constraint-based Application Model and Scheduling Techniques for Power-aware Systems, *International Symposium on Hardware/Software Codesign (CODES)*, pages 153–158, 2001.

[111] J. A. Stankovic, M. Spuri, M. D. Natale & G. C. Buttazzo, Implications of Classical Scheduling Results for Real-Time Systems, *IEEE Computer*, 28(6) pages 16–25, 1995.

[112] C. Erbas, A. D. Pimentel, M. Thompson & S. Polstra, A Framework for System-Level Modeling and Simulation of Embedded Systems Architectures, *EURASIP Journal of Embedded System*, 2007(1), pages 2–12, 2007.

[113] S. Rao, Engineering Optimization: Theory and Practice, *A Wiley-Interscience Psublication*, 1996.

[114] ILOG CPLEX, Optimization Tool, http://www.ilog.com/products/cplex/, Accessed 15 September 2008.

[115] S. Islam, Automated Specification of Automotive Software Functions Based on Expert Interviews, *Master Thesis, Institute of Industrial Automation and Software Engineering, University of Stuttgart, in cooperation with DaimlerChrysler AG, Germany*, 2004.

[116] R. L. Keeney & H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs, *Cambridge University Press*, 1993.

[117] B. Boehm, Software Engineering Economics, *Prentice-Hall*, 1981.

[118] B. Hardung, T. Kölzow & A. Krüger, Reuse of Software in Distributed Embedded Automotive Systems, *ACM International Conference on Embedded Software (EMSOFT)*, pages 203–210, 2004.

[119] S. Narayan, F. Vahid & D. Gajski, System Specification with the SpecCharts Language, *IEEE Design & Test of Computers*, 9(4), pages 6–13, 1992.

[120] OMG, Unified Modeling Language (UML) Specification, v2.1.2, http://www.omg.org/technology/documents/formal/uml.htm, Accessed 4 August 2008.

[121] The SAE Architecture Analysis and Design Language (AADL), A Standard for Model Based Engineering, http://www.aadl.info/, Accessed 17 September 2008.

[122] Architecture Description Language (ADL), http://www.sei.cmu.edu/architecture/adl.html, Accessed 17 September 2008.

[123] I. Bate & N. Audsley, Flexible Design of Complex High-Integrity Systems Using Trade Offs, *IEEE International Symposium on High Assurance Systems Engineering (HASE)*, pages 22–31, 2004.

[124] OMG: Model Driven Architecture (MDA), A Technical Perspective, *OMG Document No. ab/2001-02-04, Object Management Group*, 2003.

[125] A. Sangiovanni-Vincentelli, Defining Platform-based Design, *EEDesign of EETimes*, 2002.

[126] OMG XML Metadata Interchange (XMI), v2.0, http://www.omg.org/docs/formal/03-05-02.pdf.

[127] R. M. Keichafer, C.J. Walter, A.M. Finn & P.M. Thambidurai, The MAFT Architecture for Distributed Fault Tolerance, *IEEE Transactions on Computers*, 37(4), pages 398–405, 1988.

[128] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabl, C. Senft & R. Zainlinger, Distributed Fault-Tolerant Real-Time Systems: The Mars Approach, *IEEE Micro*, 9(1), pages 25–40, 1989.

[129] K. Alstrom & J. Torin, Future Architecture for Flight Control Systems, *Digital Avionics Systems Conference (DASC)*, 1, pages 1B5/1–1B5/10, 2001.

[130] S. Poledna, P. Barrett, A. Burns, & A. Wellings, Replica Determinism and Flexible Scheduling in Hard Real-Time Dependable Systems, *IEEE Transactions on Computers*, 49(2), pages 100–111, 2000.

[131] A. Jhumka, M. Hiller, & N. Suri, Assessing Inter-Modular Error Propagation in Distributed Software, *IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 152–161, 2001.

[132] S. Punnekkat, A. Burns & R. Davis, Analysis of Checkpointing for Real-Time Systems, *Real-Time System*, 20(1), pages 83–102, 2001.

[133] V. Izosimov, Scheduling and Optimization of Fault-Tolerant Distributed Embedded Systems, *PhD Thesis, Linköping University*, 2006.

[134] V. Izosimov, P. Pop, P. Eles & Z. Peng, Synthesis of Fault-Tolerant Embedded Systems with Checkpointing and Replication, *IEEE International Workshop on Electronic Design, Test and Applications*, pages 440–447, 2006.

[135] X. Qina & H. Jiangb, A Novel Fault-Tolerant Sheduling Algorithm for Precedence Constrained Tasks in Real-Time Heterogeneous Systems, *Journal of Parallel Computing*, 32(5-6), pages 331–356, 2006.

[136] K. Ramamritham, Allocation and Scheduling of Precedence-Related Periodic Tasks, *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 6(4), pages 412–420, 1995.

[137] P. Eles, Z. Peng, P. Pop & A. Doboli, Scheduling with Bus Access Optimization for Distributed Embedded Systems, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(5), pages 472–491, 2000.

[138] P. Miner, M. Malekpour & W. Torres, A Conceptual Design For a Reliable Optical Bus (ROBUS), *IEEE Digital Avionics Systems Conference (DASC)*, pages 1–11, 2002.

[139] J. W. S. Liu, Real-Time Systems, *Prentice Hall PTR, NJ, USA*, 2000.

[140] M. G. Harbour, M. H. Klein & J. P. Lehoczky, Timing Analysis for Fixed-Priority Scheduling of Hard Real-Time Systems, *IEEE Transactions on Software Engineering*, 20(1), pages 13–28, 1994.

[141] E. Bini & G. C. Buttazzo, Schedulability Analysis of Periodic Fixed Priority Systems *IEEE Transactions on Computers*, 53(11), pages 1462–1473, 2004.

[142] R. I. Davis, A. Zabos & A. Burns, Efficient Exact Schedulability Tests for Fixed Priority Real-Time Systems, *IEEE Transactions on Computers*, 57(9), pages 1261–1276, 2008.

[143] J. H. Anderson, J. M. Calandrino & U. C. Devi, Real-Time Scheduling on Multicore Platforms, *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 179–190, 2006.

[144] TTTech-Tools, TTTech-Tools SW Development Suite, http://www.tttech.com/products/software.htm, Accessed 14 August 2008.

[145] DECOS – Dependable Embedded Components and Systems, *IST, EU FP 6 Project*, http://www.decos.at, 2004.

[146] V. Kumar, Algorithms for Constraint-Satisfaction Problems: A Survey, *AI Magazine*, 13(1), pages 32–44, 1992.

[147] B. Hedenetz & R. Belschner, "Brake by Wire" without Mechanical Backup by Using a TTP Communication Network, *SAE World Congress*, pages 296–306, 1998.

[148] D. Prasad & J. McDermid, Dependability Evaluation using a Multi-Criteria Decision Analysis Procedure, *Proceedings of the Conference on Dependable Computing for Critical Applications (DCCA)*, pages 339–358, 1999.

[149] M. Hiller, A. Jhumka & N. Suri, An Approach for Analysing the Propagation of Data Errors in Software, *International Conference on Dependable Systems and Networks (DSN)*, pages 161–172, 2001.

[150] R. David, Random Testing of Digital Circuits: Theory and Applications, *Marcel Dekker, New York*, 1998.

[151] A. A. Ismaeel & M. A. Breuer, The Probability of Error Detection in Sequential Circuits using Random Test Vectors, *Journal of Electronic Testing: Theory and Applications*, 1(4), pages 245–256, 1991.

[152] D. Zhu, R. G. Melhem & D. Mossé, The Effects of Energy Management on Reliability in Real-Time Embedded Systems, *International Conference on Computer-Aided Design (ICCAD)*, pages 35–40, 2004.

[153] P. Czyzak & A. Jaszkiewicz, Pareto Simulated Annealing - A Metaheuristic Technique for Multiple-Objective Combinatorial Optimization, *Journal of Multi-Criteria Decision Analysis*, 7, pages 34–47, 1998.

[154] K. Smith, R. Everson & J. Fieldsend, Dominance Measures for Multi-Objective Simulated Annealing, *The Proceedings of Congress on Evolutionary Computation (CEC)*, pages 23–30, 2004.

[155] K.I. Smith, R.M. Everson, J.E. Fieldsend, C. Murphy & R. Misra, Dominance-Based Multiobjective Simulated Annealing, *IEEE Transactions on Evolutionary Computation*, 12(3), pages 323–342, 2008.

[156] S. Bandyopadhyay, S. Saha, U. Maulik & K. Deb, A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA *IEEE Transactions on Evolutionary Computation*, 12(3), pages 269–283, 2008.

[157] I. Y. Kim & O. de Weck, Adaptive Weighted Sum Method for Multiobjective Optimization: A New Method for Pareto Front Generation, *Journal of Structural and Multidisciplinary Optimization*, 31(2), pages 105–116, 2006.

[158] B. Huber, R. Obermaisser & P. Peti, MDA-Based Development in the DECOS Integrated Architecture-Modeling the Hardware Platform, *IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 43–52, 2006.

[159] A. Balogh & D. Varró, Advanced Model Transformation Language Constructs in the VIATRA2 Framework, *ACM Symposium on Applied Computing (SAC)*, pages 1280-1287, 2006, .

[160] Eclipse Foundation, http://www.eclipse.org/, Accessed 20 August 2008.

[161] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits & S. Neema, Developing Applications Using Model-Driven Design Environments, *IEEE Computer*, 39(2), pages 33–40, 2006.

[162] H. Ehrig, M. Korff & M. Löwe, Tutorial Introduction to the Algebraic Approach of Graph Grammars Based on Double and Single Pushouts, *Graph Grammars and Their Application to Computer Science, LNCS 532/1991* pages 24–37, 1991.

[163] W. Herzner, R. Schlick, M. Schlager, B. Leiner, B. Huber, A. Balogh, G. Csertán, A. LeGuennec, T. LeSergent, N. Suri & S. Islam, Model-Based Development of Distributed Embedded Real-Time Systems with the DECOS Tool-Chain, *In Proceedings of Society for Automotive Engineers (SAE) Aerotech*, 2007.

[164] DECOS Project Deliverable, PIM Design Methodology and Specification Model, http://www.decos.at, 2005.

[165] W. Herzner, A. Balogh & G. Csertán, Design Patterns for Domain-Specific Application Modeling, *Proceeding of the DECOS/ERCIM Workshop on Dependable Embedded Systems at EUROMICRO*, 2006.

[166] SCADE Suite, The Standard for the Development of Safety-Critical Embedded Software, http://www.esterel-technologies.com/products/scade-suite/, Accessed 20 August 2008.

[167] The MathWorks, The MathWorks Homepage, http://www.mathworks.com/, Accessed 21 August 2008.

[168] DO 178 Industry Group, http://www.do178site.com/, Accessed 21 August 2008.

[169] RAPID RMA: The Art of Modeling Real-Time Systems, http://www.tripac.com/html/prod-fact-rrm.html, Accessed 20 August 2008.

[170] J. A. Stankovic, VEST: A Toolset for Constructing and Analyzing Component Based Embedded Systems, *International Workshop on Embedded Software (EMSOFT)*, LNCS 2211, pages 390–402, 2001.

[171] AIRES-ToolKit, Automatic Integration of Reusable Embedded Software, http://kabru.eecs.umich.edu/bin/view/Main/AIRES, Accessed 20 August 2008.

[172] Python Software Foundation, The Python Programming Language, http://www.python.org/, Accessed 20 August 2008.

[173] DECOS Project Deliverable, Prototype of an Interactive PSM Development Tool, http://www.decos.at, 2005.

[174] DECOS Project Deliverable, Encapsulated Execution Environment (COS, POS, Tool support), http://www.decos.at, 2006.

[175] TASKING TriCore Vx-Toolset, v2.2r3, http://www.tasking.com/support/TriCore/readme_2_2r1.html, 2006.

[176] Infineon technologies; Signature Analysis Generator (SAG); User manual, Version 1.5, Dec. 2005.

[177] Lauterbach TRACE32, Microprocessor Development Tools, http://www.lauterbach.com/frames.html, 2008.

[178] J. Dongarra, E. Jeannot, E. Saule & Z. Shi, Bi-objective Scheduling Algorithms for Optimizing Makespan and Reliability on Heterogeneous Systems, *ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 280–288, 2007.

[179] O. S. Unsal & I. Koren, System-Level Power-Aware Design Techniques in Real-Time Systems, *Proceedings of the IEEE*, pages 1055–1069, 2003.

[180] Y. Zhang & K. Chakrabarty, Task Feasibility Analysis and Dynamic Voltage Scaling in Fault-Tolerant Real-Time Embedded Systems, *Design, Automation and Test in Europe Conference and Exposition (DATE)*, pages 1170–1175, 2004.

[181] Y. Cai, S. M. Reddy & B. M. Al-Hashimi, Reducing the Energy Consumption in Fault-Tolerant Distributed Embedded Systems with Time-Constraint, *International Symposium on Quality Electronic Design (ISQED)*, pages 368–373, 2007.

[182] W. Wolf, High-Performance Embedded Computing: Architectures, Applications, and Methodologies, *Morgan Kaufman*, 2006.

[183] R. Obermaisser, H. Kopetz, C. Salloum & B. Huber, Error Containment in the Time-Triggered System-On-a-Chip Architecture, *International Embedded Systems Symposium (IESS)*, 2007.

[184] R. Obermaisser, H. Kraut & C. Salloum, A Transient-Resilient System-on-a-Chip Architecture with Support for On-Chip and Off-Chip TMR, *Seventh European Dependable Computing Conference(EDCC)*, pages 123–134, 2008.

[185] C. Erbas, S. Erbas & A. D. Pimentel, Multiobjective Optimization and Evolutionary Algorithms for the Application Mapping Problem in Multiprocessor System-on-Chip Design, *IEEE Transaction on Evolutionary Computation*, 10(3), pages 358–374, 2006.

[186] W. Stechele, O. Bringmann, R. Ernst et al. Autonomic MPSoCs for Reliable Systems, *Zuverlässigkeit und Entwurf (ZuD)*, 2007.

[187] T. Streichert, D. Koch, C. Haubelt & J. Teich, Modeling and Design of Fault-Tolerant and Self-Adaptive Reconfigurable Networked Embedded Systems, *EURASIP Journal on Embedded System*, 2006(1), pages 1–15, 2006.

Index

Curriculum Vitae

2004-2008: Ph.D. (Dr.-Ing.) in Computer Science, Technische Universitt Darmstadt, Germany.

2002-2004: Master of Science (M.Sc.) in Information Technology [Focus: Embedded Systems Engineering], Universitt Stuttgart, Germany.

1995-2000: Bachelor of Science (B.Sc.) in Electrical and Electronics Engineering (EEE), Bangladesh Institute of Technology, Khulna, Bangladesh.

1982-1994: Secondary and Higher Secondary School Certification, Bangladesh.