CHAPTER 1

Introduction

1.1 Motivation

Embedded systems have penetrated our daily life without many of us noticing, by that blurring what we mean by an embedded system. Our day-to-day routine gets into contact with computer systems in all aspects, and some of these daily encounters depend on the correctness of embedded computers. Modern, public transportation systems offer a fully automated, unattended train operation which is capable of handling starting, stopping, door operation as well as emergency situations. Similarly, modern cars are equipped with semi-automatic driver assistance features and it is only a matter of time until autonomous driving will be the common case.

Medical devices such as wearable health technology are predicted to revolutionize medical care. Gadgets for medical therapy, sports, or just every-day fitness are capable to track brainwaves, heart rate, blood glucose level, sleep pattern, and more. Wearable devices cannot only be used to monitor and track but also to regulate for instance inject medication or stimulate nerve cells. It is predicted that by end of 2016 more than 100 million wearable medical devices are sold per year. The market for fitness related products will reach 80 million units by then.

In the context of embedded systems, *safety critical systems* play an important role in medical care, commercial aircraft, nuclear power, and weapons [153]. There are many different definitions of what safety critical precisely means. A customary meaning is given in [153] which encompasses

> "systems whose failure might endanger human life, lead to substantial economic loss or cause extensive environmental damage."

This is consistent with most readers intuition, which account railway signaling systems, flight control systems as well as steer by wire as safety critical. However, a more general definition is also given by [153] which matches the one of [13] which is based on the notion of consequences.

> "If the failure of a system could lead to consequences that are determined to be unacceptable, then the system is safety-critical."

Traditionally, safety critical systems were closed, self-contained computers systems with very limited interface to its environment. This includes systems such as the Ariane 5 rocket of which a crash can result in a financial loss of more than US$ 370 million (Cluster spacecraft incident) as well as the Boeing 777 which is equipped with several computerized systems which replaced most of the traditional mechanical and hydraulic equipment. A report by the National Transportation Safety Board to the Federal Aviation Administration (FAA) [235] describes serious problems with the glass cockpit displays which replaced the traditional analog dials and gauges. These problems have led to at least 50 in-flight incidents, some of these causing the pilots to panic due to blank displays and lost communication. Such a failure can result in the death of hundreds of passengers.

However, recently a new specimen of non-traditional safety-critical systems has emerged. Such systems are not directly linked to catastrophic hazards, but may indirectly cause them. Nowadays, the cellular phone network does not only provide a convenient way to communicate with each other, but is also the backbone for emergency service (i.e. 112/911). In most countries the cellular network serves a dual use: it is used to signal an emergency to authorities as well by the authorities themselves, mainly to coordinate the operation. The importance of the cellular infrastructure for the greater public good (saving lives, preventing fires, etc.) elevates the former convenience technology to a safety-critical level. Other non-traditional sectors include banking, (non-nuclear) electricity generation, management of water systems (i.e. desalination).

1.2 The Role of Safety Standards

In the last years, we saw a strong trend towards standardization of the entire safety life cycle. Traditional quality assurance and process management guidelines such as ISO 15504 / SPICE [139] or ISO 9001 [138] are not suitable for the development process of safety critical systems.

This is already known from the conservative avionics industry, in which software must be developed and tested according to the domain specific standard DO-178b [224] and hardware components according to DO-254 [223]. The final aircraft will only achieve FAA approval (Type Certificate), if the

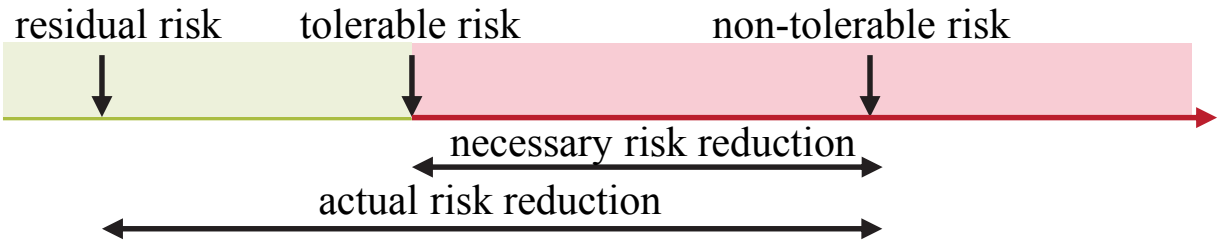

Figure 1.1: If the risk is not tolerable, additional measures such as fault-tolerance must be applied.

rules and processes of the required standards are obeyed. A similar process is compulsory for industrial plants such as power plants and heavy machinery [135].

Interestingly, such standards are rather new to the automotive domain and were not introduced prior to 2011. This has two reasons: Firstly because the consequences of a car crash are mostly considered as benign compared to a plane crash and secondly, because automotive manufacturers were keen to provide very high quality products to prevent liability issues. This changed with the introduction of the ISO 26262 [136] which is loosely based on its industrial counterpart IEC 61508. However, there are some differences. IEC 61508 is targeted towards equipment produced in low quantities, where ISO 26262 addresses volume production of the automotive market. Since then, industry puts a tremendous effort into developing a safety culture around their products.

The concept around ISO 26262 is based on risk and, as previously explained, safety is defined as the absence of unreasonable risk. Although the concept of risk seems to be very obvious, it is rather complicated to asses and systematically biased by the limitation of the human mind. The human mind tends to apply simple heuristics when risk is assessed. People are bias towards recent news and experiences which leads to a cognitive bias towards these events. This is called *availability heuristic* [277]. In order to systematically assess risk, the combination of likelihood of occurrence and the severity of the harm of a hazard must be considered. The risk is tolerable if society can accept it and safety standards guide the designer to determine and quantify the acceptance.

Figure 1.1 shows the typical case for a safety critical system. After a particular function is evaluated according to the guidelines dictated by the standard, it is evident that the risk is non-tolerable. This can be the case if standard implementations such as commercial of the shelf (COTS) hardware or software are too error prone. Thus, the actual risk which emanates from the function must be reduced by applying further measures covered for instance by using a different technology or fault tolerance

approaches. Any deployed function which is integrated in a larger system obviously still exhibits a residual risk - but safety standards guarantee that this risk remains below a tolerable threshold.

Now, it is interesting to know the additional effort required to be compliant with the state of the art safety standards, their methods and processes [244].

- Generally, risk dictates effort.
- Comparison with reference products is required.
- Assessment of known information and data must be carried out.
- Additional research is required for novel features with high risk which do not origin from previously used ancestor technology.

When designing a traditional safety critical system, the entire system context must be known. This includes the platform architecture, deployed software modules and their interaction as well as the physical boundary conditions such as worst-case environmental conditions (i.e. vibration, temperature and other stress).

1.3 Development Process for Safety-Critical Systems

To handle and master a successful safety critical embedded system, an appropriate development process is mandatory [123]. The automotive industry, especially in Germany, typically applies the V-Model [2, 247, 136].

The V-Model separates the design and specification from implementation and testing as shown in Figure 1.2. Safety standards such as the ISO 26262 have refined the V-Model and incorporated the safety requirements and safety verification into the process. This ensures traceable level of design complexity and intrinsically produces the required assurance level required by the certification agency.

Contrarily to the standard V-Model as described in [2], the V-Model as used in ISO 26262 starts with a safety assessment as a starting point. Here safety functions are identified, the risk is assessed and a high level functional safety concept is produced. The safety concept is formalized as a safety requirements specification which is later used for the functional safety assessment, to validate whether the final system satisfies all safety concerns. This typically involves a Fault Tree Analysis (FTA) [134], which is a top-down failure analysis that reveals the root cause for undesirable or catastrophic events which can be linked to the system under design [171].

In the system design step, the system architecture is specified and broken down into components with specified interfaces. This includes the hardware architecture such as communication and processing platforms

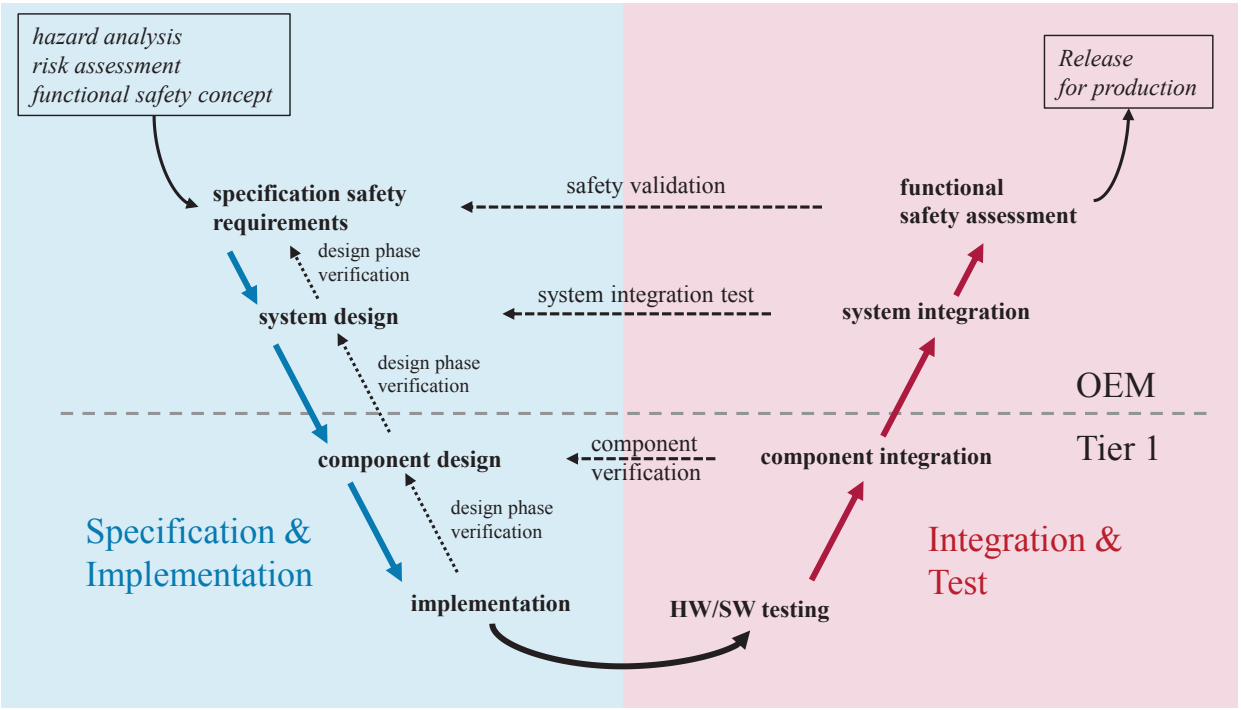

Figure 1.2: Simplified V-Model according to ISO 26262 [136].

as well as the high level software architecture. Here, safety standards recommend to capture a consistent set of requirements for instance by using Controlled Requirements Expression (CORE)[188]. Furthermore, tools, models and languages to reflect functional and non-functional behavior are strongly advised (e.g. MARTE[203], MATLAB / Simulink, AADL[238], SysML[204]).

In the component design process, individual components are broken down into function blocks which are later implemented by a programmer or hardware designer (bottom of Figure 1.2). For the hardware and software specification, ISO 26262 demands a continuous evaluation on the impact on safety. For instance, once the hardware platform is known, fault injection tests and further reliability tests should be carried out. Otherwise, there is an unknown risk of exceeding the reliability threshold and missing the safety goals. These failure tests are performed inline with test automation such as hardware-in-the-loop (HIL) tests, rest-bus simulations.

The right branch of the V-Model, Integration & Test, is responsible to verify and test the implemented functions and components against the specification. Naturally, this includes the error-free behavior as well as the service in case of errors. When the final safety validation step is completed successfully, the system can be released for production.

As shown in Figure 1.2, the component design and implementation is usually performed by the suppliers. To ease this transition step, the auto-

motive industry has standardized to automotive software and operating system interfaces in scope of the Automotive Open Systems Architecture (AUTOSAR) [12]. The concept of AUTOSAR is to focus on portability, composability and extendability where possible. Here AUTOSAR specifies a Runtime Environment which provides platform as well as communication abstraction for applications [116]. It implements well defined interfaces to connect external communication interfaces such as FLEXRAY, Controller Area Network, Ethernet and others.

AUTOSAR follows a component based design approach in which functions are encapsulated in AUTOSAR Software Components (SW-C). These components have well defined interfaces according to a standard description format. Software components are connected to a virtual function bus which abstracts the physical communication technology and allows application agnostic message passing. This allows an easy cut of system functionality into components without large overhead while maintaining a high degree of flexibility.

1.4 Trends

The industry impact of embedded systems has increased during the last decades and this trend is predicted to continue. The reason for this is that embedded computing and electronics are the main driver for features and the key for product differentiation. According to [217], the embedded systems market will reach a €1.5 trillion in revenue by 2015. The most important market segments measured by their compound annual growth rate (CAGR) are energy (45.4%), communications (13.2%), automotive (12.2%) as well as healthcare (11.4%). Interestingly the growth of consumer products is predicted to be only 6.2%. This highlights the importance of the special requirements and constraints of highly specialized domains with unique constraints such as low energy, low cost, ultra-high reliability, hard real-time under extreme environmental conditions. These market segments have to tackle the following new challenges to continue successfully their growth.

1.4.1 Architecture Complexity Challenge

There is a rapid technology advancement which enables the designer to add more and more features and functionality to the system. As a consequence, the size and complexity grows exponentially. This problem is likely getting worse, if the additional complexity cannot be conquered by compositional model-based design processes. Generally, there are two orthogonal dimensions to the complexity challenge: architecture as well as software complexity. The software complexity for a system in the automotive domain

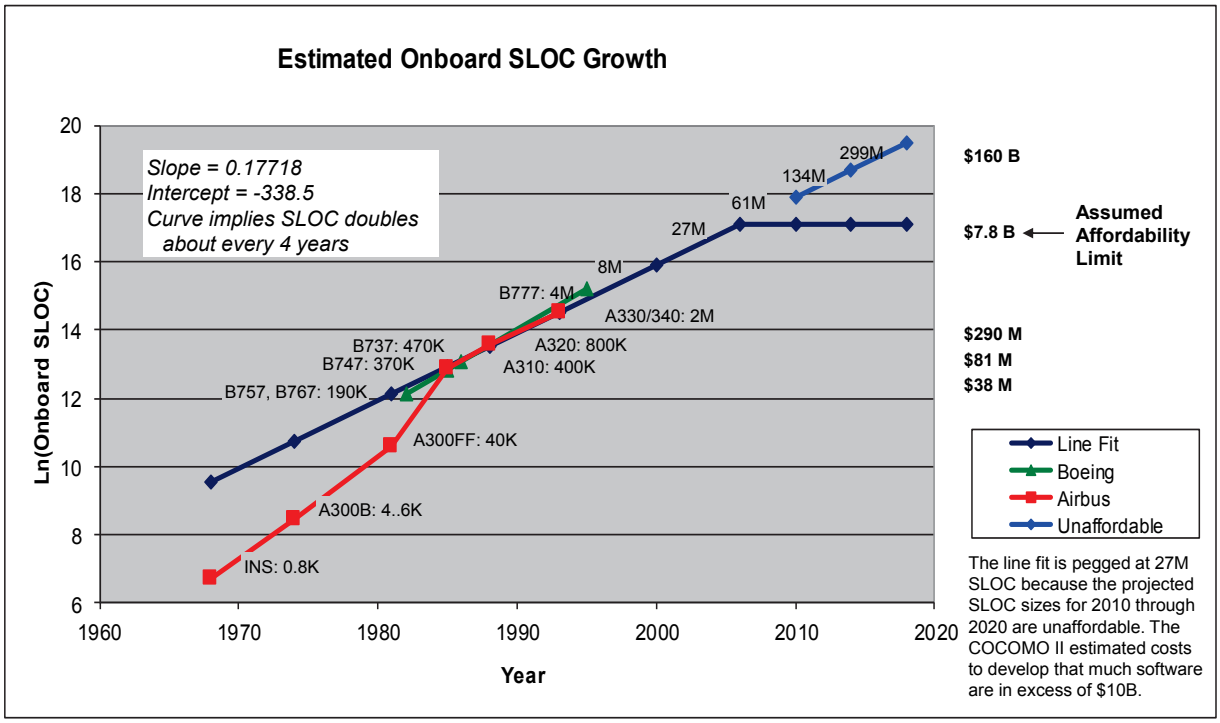

Figure 1.3: Amount of certified software code and the associated cost. (Source [285])

increased by two orders of magnitude (10^6 to 10^8 object instructions) in only 10 years which is comparable to the growth of the linux kernel during the same timespan [83]. A similar trend can be observed in the avionics industry [285], where the code size roughly doubles every year. It was estimated that by 2008 the associated costs including certification according to safety standards exceed a $ 7.8 billion threshold. This is assumed to be the affordability limit, software which exceeds 17 million lines of code is predicted to be uneconomical for aircraft designs. A modern A380 aircraft already has 100 million lines of code [287]. Handling this enormous complexity was only possible by applying formal methods such as model checking, model driven engineering on platform level as well as a compositional analysis on system level. Also standardized and modular software architectures ease the design process. Examples for such frameworks are AUTOSAR [12] used in the automotive domain as well as ARINC 653 [6] which is used in avionics.

Also the hardware platforms become more and more powerful. This advancement has boosted the data rate and processing performance required for today's and tomorrow's advanced driver assistance. Typical examples are in-vehicle navigation systems, adaptive cruise control and sophisticated camera-based precrash detection systems. The integration of multiple

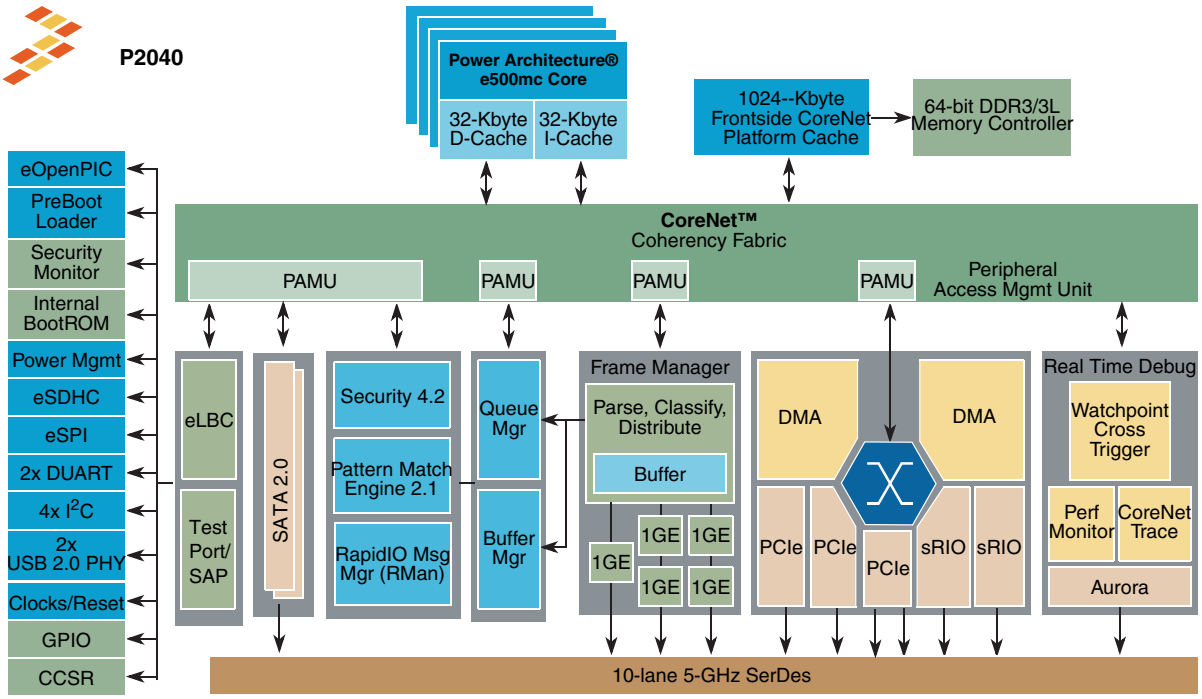

Figure 1.4: Block diagram of Freescale P2040 multi-core processor (Source [97])

processing elements allows to reduce the frequency, thus power and temperature and are an attractive design target for all computing domains. At first glance, this sounds promising as it enables the integration of tremendous complexity in the first place. The vast number of cores can potentially be used to integrate and partially isolate different functionality on such a platform. However, there is a downside: Processing elements found on multi- as well as many-core architectures share many common resources such as the communication infrastructure, caches, memory controllers and I/O ports. An example for a modern multi-core architecture is shown in Figure 1.4, a switch fabric connects all cores to shared DMA units, shared platform cache, a single DDR3 memory controller and various peripherals. This causes an easily overlooked entanglement of the timing and performance for the applications running on the platform [159]. Obviously, this inhibits a straight forward compositional consideration and leads to additional complexity during the verification stage. Compared to traditional architectures, the behavior of multi-core designs seems unpredictable and afflicted with complex to grasp timing anomalies. Therefore, traditional design processes are not applicable to multi- and many-core designs. Also, recent research has shown that the real-time performance of multi-core architectures does not necessarily outperform traditional single core designs [27] in all cases.

1.4.2 Cyber-Physical Systems Challenge

Most systems that we know today such as traffic control, health care, automotive safety, smart power grids, defense systems, environmental control and manufacturing have a tight coupling of computing and the networking infrastructure with physical processes. These systems are an integral part of the feedback loop where the physical processes affect computations and the other way around [169]. Sometime in 2008, the name cyber-physical system (CPS) was coined and serious research in this domain just started a couple of years ago.

In the physical world, the passage of time is inevitable and processes (e.g. mechanical, chemical) are concurrent by nature. Contrarily to the physical world, computation and communication models are intrinsically sequential and lack the proper abstraction. Timing and predictability was often neglected in computer science as pipeline design, caches and compiler design was tweaked to optimize the average-case performance ("make the common case fast").

A new level of abstraction must be found [26] which effectively combines computational models with models of the physical process to properly capture mutual dependencies. Here, traditional software component technologies failed as they are too software centristic. This includes operating system design, object oriented programming and service oriented architectures, because they abstract away important part of the system behavior (i.e. timing) as they try to focus only on the functional aspect of component design.

1.4.3 System of Systems Challenge

As discussed, new markets emerged such as smart electricity and water meters used for monitoring which will boost the sales of low-power, low-cost hardware. The next step is to combine embedded systems in a large scale global network of data and services. This leads to a new situation [60]: Systems of Systems (SoS) with a world of high computing density and drastically increased data rates and traffic volumes. There is no generally accepted definition for Systems of Systems. However, it is common ground that SoS are

> "themselves comprised of multiple autonomous embedded complex systems that can be diverse in technology, context, operation, geography and conceptual frame." [149].

An example for a typical SoS is the Coast Guard Deepwater Program [206] which is a 25 year program that connects recovery aircrafts, patrol boats, unmanned aerial vehicles with ground stations such as command, control and intelligence to replace almost all of today's US Coast Guard's

equipment. Other examples are FAA Air Traffic Management, Army Future Combat Systems, intelligent transport systems as well as Robotic Colonies.

In scope of the United States national space program new System of Systems engineering models and frameworks were proposed [65], which are now being adopted for non-defence related projects. These frameworks do not only account for the technological challenges but also consider the political, social and economic factors. SoS ultimately lead to heterogeneous, distributed architectures and it remains to be seen if such complex systems can be still be realized, validated and handled, if this trend continues.

1.4.4 Adaptability and Software Evolution Challenge

Today, a typical automotive vehicle design comprises many electronic control units each implementing distinct functionality (e.g. anti-lock braking unit, traction control system, emergency break assist). An upgrade of functionality is only possible through facelift upgrading or a completely new car design. Upgrading an deployed car is cumbersome and expensive: For example, the latest engine management configuration cannot be integrated without an expensive recall. On the other hand, customers have high expectations with respect to the in-vehicle infotainment system. They are used to the update cycle of entertainment products in the order of a few month. Google deploys major updates for their smartphone operating system Android every six to nine months where automotive entertainment software is never updated at all unless the customer decides to buy a new car, typically after five to six years.

Also, there is a paradigm shift towards software and network centric automotive design. New features in the automotive industry are mostly software driven and could be retrofitted into legacy cars. Such an “app store” opens up a totally new business model for OEMs and dealers.

However, the concept of software adaptability and evolution is not completely new [88]. But it has never been considered in the context of embedded as well as cyber physical system, where adaptability is inevitably linked with two conceptual problems: The *first challenge* is the competition of applications for resources. New applications share the same platform, this includes the communication infrastructure such as busses and switches as well as processors and memory. And the *second challenge* is the impact of platform and architecture change. If new hardware is added (i.e. a head up display is added to the system), other devices must be aware of the new functionality (i.e. for signal routing and configuration).

Both effects tightly couple legacy and new functionality. This is extremely problematic in domains where safety, security and availability are key constraints as such properties cannot easily be guaranteed after platform or software changes. Especially when new functionality cannot be trusted because it is developed by an unknown supplier. Novel mechanisms

which must be provided at design time must guarantee sufficient isolation, while not sacrificing flexibility and performance. Examples for such dynamic methods are load balancing, integration of quality of service and dynamic resource management [5]. Additionally these approaches must be integrated to provide feedback-based resource scheduling, middleware support for dynamic updates and new dynamic models which can be used for on-line verification. Otherwise integrity cannot be preserved and new subsystems cannot be admitted.

1.4.5 Resiliency Challenge

Continuous growth of complexity always reflects on the reliability of a system caused by nature of statistics. Interestingly, this is not a new phenomenon in computer technology. In the past, the cause for faults used to be the manufacturing and development process that impacted the quality of a product. This was tackled by testing the circuits and sorting out bad ones. Also the environment in which the device is operated affects the reliability. For instance, the soft-error rate increases with altitude. Future semiconductor devices will face new challenges [43, 36]:

- Transistor variability
- Device degradation
- Sensitivity to ionizing rays and particles

This leads to reliability problems of modern and future silicon devices which is illustrated in Figure 1.5. The graph shows the quality (i.e. speed grade) of a silicon gate over the time. Each dot represents an instance of the gate over time. After manufacturing, some devices are faster than others due to process variability. Thus, some gates are beyond the acceptable quality threshold (red area). Over time, aging effects lead to consistent decrease of performance. After some time, the gate operates out of the specified operating conditions. Also spontaneous, transient effects can occur (e.g. caused by negative-bias temperature instability). Generally, these effects are inherent to the silicon process and already existed in previous generations. However, in next generation devices these characteristics will appear much more pronounced.

The size of a transistor will, if the trend continues, decrease further - even beyond todays (2014) 22 nm technology node. At this stage, various effects become noticeable [184]. The feature sizes will be so small that different dopant areas will be separated by only a few atom layers. This causes dopant fluctuation which comes from the discreteness of dopant atoms in the transistor channel. Thus, because the law of large numbers