Chapter 1
Introduction

This chapter presents terms used in the context of embedded systems together with their history as well as opportunities, challenges, and common characteristics of embedded and cyber-physical systems. Furthermore, educational aspects, design flows, and the structure of this book are introduced.

1.1 History of Terms

Until the late 1980s, information processing was associated with large mainframe computers and huge tape drives. Later, miniaturization allowed information processing with personal computers (PCs). Office applications were dominating, but some computers were also controlling the physical environment, typically in the form of some feedback loop.

Later, Mark Weiser created the term "ubiquitous computing" [573]. This term reflects Weiser's prediction to have computing (and information) *anytime, anywhere*. Weiser also predicted that computers are going to be integrated into products such that they will become invisible. Hence, he created the term "invisible computer." With a similar vision, the predicted penetration of our day-to-day life with computing devices led to the terms "pervasive computing" and "ambient intelligence." These three terms focus on only slightly different aspects of future information technology. Ubiquitous computing focuses more on the long-term goal of providing information anytime, anywhere, whereas pervasive computing focuses more on practical aspects and the exploitation of already available technology. For ambient intelligence, there is some emphasis on communication technology in future homes and smart buildings. Due to the widespread use of small devices in combination with the mobile Internet, some of the visions about the future have already become a common practice. This widespread use is pervasive in the sense

P. Marwedel, *Embedded System Design*, Embedded Systems,
https://doi.org/10.1007/978-3-030-60910-8_1

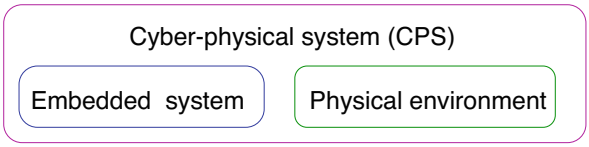

Fig. 1.1 Relationship between embedded systems and CPS

that it already had an impact on many areas of our life. Furthermore, artificial intelligence is influencing our life as well.

Miniaturization also enabled the integration of information processing and the environment using computers. This type of information processing has been called an "embedded system":

Definition 1.1 (Marwedel [371]) "*Embedded systems are information processing systems embedded into enclosing products*."

Examples include embedded systems in cars, trains, planes, and telecommunication or fabrication equipment. Embedded system products such as self-driving cars and trains are already available or have been announced. Consequently, we can expect miniaturization to have an impact on embedded systems comparable to the one it had on the availability of mobile devices. Embedded systems come with a large number of common characteristics, including real-time constraints, and dependability as well as efficiency requirements. For such systems, the link to physical systems is rather important. This link is emphasized in the following citation [331]:

"*Embedded software is software integrated with physical processes. The technical problem is managing time and concurrency in computational systems.*"
This citation could be used as a definition of the term "embedded software" and could be extended into a definition of "embedded systems" by just replacing "software" by "system."

However, the strong link to physical systems has recently been stressed even more by the introduction of the term "cyber-physical systems" (CPS for short). CPS can be defined as follows:

Definition 1.2 (Lee [332]) "*Cyber-Physical Systems (CPS) are integrations of computation and physical processes*."

The new term emphasizes the link to physical processes and the corresponding physical environment. Emphasizing this link makes sense, since it is frequently ignored in a world of applications running on servers, PCs, and mobile phones. For CPS, models should include models of the physical environment as well. The term CPS comprises an embedded system (the information processing part) and a (dynamic) physical environment or **CPS = ES + (dynamic) physical environment**. This is also reflected in Fig. 1.1.

In their call for proposals, the National Science Foundation in the USA mentions also communication [412]: "*Emerging CPS will be coordinated, distributed, and **connected** and must be robust and responsive*."

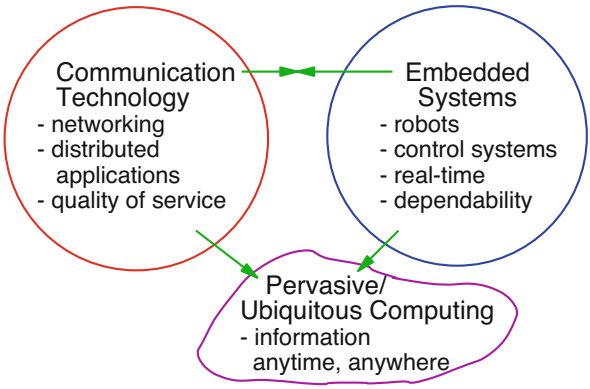

Fig. 1.2 Importance of communication (© European Commission)

This is also done in the acatech report on CPS [6]: CPS . . . "*represent **networked**, software-intensive embedded systems in a control loop, provide networked and distributed services*."

Interconnection and collaboration are also explicitly mentioned in a call for proposals by the European Commission [155]: "*Cyber-Physical Systems (CPS) refer to next generation embedded ICT systems that are **interconnected** and collaborating including through the **Internet of Things**, and providing citizens and businesses with a wide range of innovative applications and services*."

The importance of communication was visualized by the European Commission earlier, as shown in Fig. 1.2.

From these citations, it is clear that the authors do not only associate the integration of the cyber- and the physical world with the term CPS. Rather, there is also a strong communication aspect. Actually, the term CPS is not always used consistently. Some authors emphasize the integration with the physical environment, others emphasize communication.

Communication is more explicit in the term "Internet of Things" (IoT), which can be defined as follows:

Definition 1.3 ([185]) The term Internet of Things "*describes the pervasive presence of a variety of devices — such as sensors, actuators, and mobile phones — which, through unique addressing schemes, are able to interact and cooperate with each other to reach common goals*."

This term is linking sensors (such that sensed information is available on the Internet) and actuators (such that things can be controlled from the Internet). The Internet of Things is expected to allow the communication between trillions of devices in the world. This vision affects a large amount of businesses.

The exploitation of IoT-technology for production has been called "Industry 4.0" [68]. Industry 4.0 targets a more flexible production for which the entire life cycle from the design phase onward is supported by the IoT.

To some extent, it is a matter of preferences whether the linking of physical objects to the cyber-world is called CPS or IoT. Taken together, CPS and IoT include most of the future applications of IT.

The design of these future applications requires knowing fundamental design techniques for embedded systems. This book focuses on such fundamental techniques and on foundations of embedded system design. Please remember that these are used in IoT and CPS designs though this is not repeatedly stated in each context. However, application-specific aspects of CPS and IoT are usually not covered.

1.2 Opportunities

There is a huge potential for applications of information processing in the context of CPS and IoT. The following list demonstrates this potential and the large variation of corresponding areas:

- **Transportation and mobility:**
  - **Automotive electronics:** Modern cars can be sold in technologically advanced countries only if they contain a significant amount of electronics [415]. These include airbag control systems, engine control systems, navigation systems, anti-braking systems (ABS), electronic stability programs (ESP), air-conditioning, anti-theft protection, driver assistance systems, and many more. There is a trend toward autonomous driving. Embedded systems can improve comfort levels, avoid accidents, and reduce the impact on the environment. E-mobility would not be feasible without a significant amount of electronic components.
  - **Avionics:** A significant amount of the total value of airplanes is due to the information processing equipment, including flight control systems, anti-collision systems, pilot information systems, autopilots, and others. Dependability is of utmost importance.[1] Embedded systems can decrease emissions (such as carbon dioxide) from airplanes. Autonomous flying is also becoming a reality, at least for certain application areas.
  - **Railways:** For railways, the situation is similar to the one discussed for cars and airplanes. Again, safety features contribute significantly to the total value of trains, and dependability is extremely important. Advanced signaling aims at safe operation of trains at high speed and short intervals between trains. The European Train Control System (ETCS) [444] is one step in this direction. Autonomous rail-based transportation is already used in restricted contexts like shuttle trains at airports.

[1] Problems with Boeing's 737 MAX [419] underline this statement.

  - **Maritime engineering** (ships, ocean technology, and other maritime systems)**:** Maritime systems, such as modern ships, use large amounts of ICT equipment, e.g., for navigation, for safety, for optimizing the operation in general, and for bookkeeping (see, e.g., http://www.smtcsingapore.com/ and https://dupress.deloitte.com/dup-us-en/focus/internet-of-things/iot-in-shipping-industry.html).
  - **New concepts for mobility:** The use of ICT technology and its components is enabling new concepts for mobility. Even untrained people can travel larger distances with e-bikes. The subtle interaction between human muscles and electric engines turns e-scooters into a prime example of cyber-physical systems. The collection of e-scooters at the end of each day, based on a list of locations in the Internet, lets e-scooters become a perfect example of the Internet of Things. Also, CPS/IoT-technology is very important for collective taxis and other taxi-calling services.

- **Mechanical engineering** (incl. manufacturing)**:** Machinery and fabrication equipment have been combined with embedded systems for decades. In order to optimize production technologies further, CPS/IoT-technology can be used. CPS/IoT-technology is the key toward more flexible manufacturing, being the target of "Industry 4.0" [68]. Factory automation is enabled by logistics. There are several ways in which CPS/IoT-systems can be applied to **logistics** [297]. For example, radio frequency identification (RFID) technology, if used in combination with computer networks, provides easy identification of each and every object, worldwide. Mobile communication allows unprecedented interaction.
- **Robotics:** This is also a traditional area in which embedded/cyber-physical systems have been used. Mechanical aspects are very important for robots. Hence, they may be linked to mechanical engineering. Robots, modeled after animals or human beings, have been designed. Figure 1.3 shows such a robot.
- **Power engineering and the smart grid:** In the future, the production of energy is supposed to be much more decentralized than in the past. Providing stability in such a scenario is difficult. ICT technology is required in order to achieve a sufficiently stable system. Information on the smart grid can be found, for example, at https://www.smartgrid.gov/the_smart_grid and at http://www.smartgrids.eu/.
- **Civil engineering:** CPS devices can be beneficial in many applications of civil engineering. This includes structural health monitoring. Natural and artificial structures like mountains, volcanoes, bridges, and dams (see, e.g., Fig. 1.4) are potentially threatening lives. We can use embedded system technology to enable advance warnings in case of increased dangers like avalanches or collapsing dams.[2]

[2]The case of the dam in Brumadinho (see https://en.wikipedia.org/wiki/Brumadinho_dam_disaster) is a counterexample of how modern sensors should be exploited.

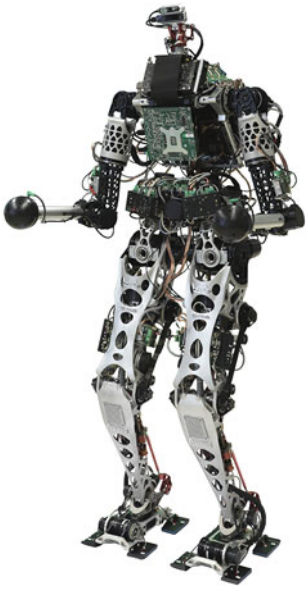

Fig. 1.3 Humanoid Robot "Lola", © Chair of Applied Mechanics, Technical University of Munich (TUM)

Fig. 1.4 Example of a dam to be monitored (Möhnesee dam), ©P. Marwedel

- **Disaster recovery:** In the case of major disasters such as earthquakes or flooding, it is essential to save lives and provide relief to survivors. Flexible communication infrastructures are essential for this.
- **Smart buildings:** Smart buildings are one of the areas of civil engineering. Information processing can be used to increase the comfort level in buildings, can reduce the energy consumption within buildings, and can improve safety and security. Subsystems which traditionally were unrelated must be connected for this purpose. There is a trend toward integrating air-conditioning, lighting, access control, accounting, safety features, and distribution of information into a single system. Tolerance levels of air-conditioning subsystems can be increased for

empty rooms, and the lighting can be automatically reduced. Air-condition noise can be reduced to a level required for the actual operating conditions. Intelligent usage of blinds can also optimize lighting and air-conditioning. Available rooms can be displayed at appropriate places, simplifying ad hoc meetings and cleaning. Lists of non-empty rooms can be displayed at the entrance of the building in emergency situations (provided the required power is still available). This way, energy can be saved on cooling, heating, and lighting. Also safety can be improved. Initially, such systems might mostly be present in high-tech office buildings, but the trend toward energy-efficient buildings also affects the design of private homes. One of the goals is to design so-called zero-energy-buildings (buildings which produce as much energy as they consume) [426]. Such a design would be one contribution toward a reduction of the global carbon-dioxide footprint and global warming.

- **Agricultural engineering:** There are many agricultural applications. For example, the "*regulations for traceability*[3] *of agricultural animals and their movements require the use of technologies like IoT, making possible the real time detection of animals, for example, during outbreaks of (a) contagious disease*" [516].
- **Health sector** and **medical engineering:** The importance of healthcare products is increasing, in particular in aging societies. Opportunities start with new sensors, detecting diseases faster and more reliably. New **data analysis** techniques (e.g., based on machine learning) can be used to detect increased risks and improve chances for healing. Therapies can be supported with personalized medication based on artificial intelligence methods. New devices can be designed to help patients, e.g., handicapped patients. Also, surgery can be supported with new devices. Embedded system technologies also allow for a significantly improved result monitoring, giving doctors much better means for checking whether or not a certain treatment has a positive impact. This monitoring also applies to remotely located patients. Available information can be stored in patient information systems. Lists of projects in this area can be found, for example, at http://cps-vo.org/group/medical-cps and at http://www.nano-tera.ch/program/health.html.
- **Scientific experiments:** Many contemporary experiments in sciences, in particular in physics, require the observation of experiment outcomes with IT devices. The combination of physical experiments and IT devices can be seen as a special case of CPS.
- **Public safety:** The interest in various kinds of safety is also increasing. Embedded and cyber-physical systems and the Internet of Things can be used to improve safety in many ways. This includes public health in times of pandemics and the identification/authentication of people, for example, with fingerprint sensors or face recognition systems.

[3]The importance of traceability in general, beyond animals, became particularly obvious during the Corona-19 crisis.

- **Military applications:** Information processing has been used in military equipment for many years. Some of the first computers analyzed military radar signals.
- **Telecommunication:** Mobile phones have been one of the fastest-growing markets in the recent years. For mobile phones, radio frequency (RF) design, digital signal processing, and low-power design are key aspects. Telecommunication is a salient feature of IoT. Other forms of telecommunication are also important.
- **Consumer electronics:** Video and audio equipment is a major sector of the electronics industry. The information processing integrated into such equipment is steadily growing. New services and better quality are implemented using advanced digital signal processing techniques. Many TV sets (in particular high-definition TV sets), multimedia phones, and game consoles comprise powerful high-performance processors and memory systems. They represent special cases of embedded systems. Compared to other types of embedded systems, safety and real-time behavior are less important. Nevertheless, certain real-time constraints must be met in order to achieve a certain frame rate or to meet time constraints for communication protocols. Also, there is a limited availability of resources like electrical energy and communication bandwidth. In this sense, limited availability of resources is a feature which consumer electronics shares with the other application areas mentioned so far.

The large set of examples demonstrates the huge variety of applications of embedded systems in CPS and IoT systems. Even more applications are listed in a report on opportunities and challenges of the IoT [516]. In a way, many of the future applications of ICT technology can be linked to such systems. From the above list, we conclude that **almost all engineering disciplines will be affected**.

The long list of application areas of embedded systems is resulting in a corresponding economic importance of such systems. The acatech report [6] mentions that, at the time of writing the report, 98% of all microprocessors were used in these systems. In a way, embedded system design is an enabler for many products and has an impact on the combined market volume size of all the areas mentioned. However, it is difficult to quantify the size of the CPS/IoT market since the total market volume of all these areas is significantly larger than the market volume of their ICT components. Referring to the value of semiconductors in the CPS/IoT market would also be misleading, since that value is only a fraction of the overall value.

The economic importance of CPS and the IoT is reflected in calls for proposals by funding organizations, like the NSF [116] and the European Commission [156].

1.3 Challenges

Unfortunately, the design of embedded systems and their integration in CPS and IoT systems comes with a large number of difficult design issues. Commonly found issues include the following:

- Cyber-physical and IoT systems must be **dependable**.

  Definition 1.4 A system is **dependable** if it provides its intended service with a high probability and does not cause any harm.

  A key reason for the need of being dependable is that these systems are directly connected to the physical environment and have an immediate impact on that environment. The issue needs to be considered during the entire design process.

  Dependability encompasses the following aspects of a system:

1. **Security:**

   Definition 1.5 ([75, 255]) Information security can be defined as the "*preservation of confidentiality, integrity and availability of information.*"

   This preservation can be compromised by thefts or damages, resulting from attacks from the **outside**. Connecting components in IoT systems enables such attacks, with cyber-crime and cyber-warfare as special, potentially harmful cases. Connecting more components enables more attacks and more damages. This is a serious issue in the design and proliferation of IoT systems.

   The only really secure solution is to disconnect components, which contradicts the idea of using connected systems. Related research is therefore expected to be one of the fastest-growing areas in ICT-related research.

   According to Ravi et al. [300], the following typical elements of security requirements exist:

   – A **user identification process** validates identities before allowing users to access the system.
   – **Secure network access** provides a network connection or service access only if the device is authorized.
   – **Secure communications** include a number of communication features.
   – **Secure storage** requires confidentiality and integrity of data.
   – **Content security** enforces usage restrictions.

2. **Confidentiality** is one of the aspects of security .

   Definition 1.6 ([255]) Confidentiality is "*property that information is not made available or disclosed to unauthorized individuals, entities, or processes.*"

   Confidentiality is typically implemented using techniques which are found in secure systems, e.g., encryption.

3. **Safety:**

   Definition 1.7 ([250]) Safety can be defined as the absence of "*unacceptable risk of physical injury or of damage to the health of people, either directly or indirectly as a result of damage to property or to the environment.*"

   "***Functional safety** is the part of the overall safety that depends on a system or equipment operating correctly in response to its inputs.*"

"*In the context of computer systems, this term is used to distinguish from threats due to external attacks, e.g., due to malicious software. In contrast to such threats, safety refers to risks caused by failures occurring without any external action, e.g., hardware failures, power failures, incorrectly written software, or operator errors*" (translated from German [576]).

4. **Reliability:** This term refers to malfunctions of systems resulting from components not working according to their specification at design time. Lack of reliability can be caused by breaking components. Reliability is the probability that a system will not fail within a certain amount of time.[4] For an evaluation of reliability, we are not considering malicious attacks from the outside but only effects occurring within the system itself during normal, intended operation.
5. **Repairability:** Repairability (also spelled reparability) is the probability that a failing system can be repaired within a certain time.
6. **Availability:** Availability is the probability that the system is available. Reliability and repairability must be high and security hazards absent in order to achieve a high availability.

Designers may be tempted to focus just on the functionality of systems initially, assuming that dependability can be added once the design is working. Typically, this approach does not work, since certain design decisions will not allow achieving the required dependability in the aftermath. For example, if the physical partitioning is done in the wrong way, redundancy may be impossible. Therefore, *"making the system dependable must not be an after-thought"*, it must be considered from the very beginning [303]. Good compromises achieving an acceptable level of safety, security, confidentiality, and reliability have to be found [296].

Even perfectly designed systems can fail if the assumptions about the workload and possible errors turn out to be wrong [303]. For example, a system might fail if it is operated outside the initially assumed temperature range.

- If we look closely at the interface between the physical and the cyber-world, we observe a **mismatch between physical and cyber models**. The following list shows examples:
  - Many cyber-physical systems must meet **real-time constraints**. Not completing computations within a given time frame can result in a serious loss of the quality provided by the system (e.g., if the audio or video quality is affected) or may cause harm to the user (e.g., if cars, trains, or airplanes do not operate in the predicted way). Some time constraints are called hard time constraints:

    Definition 1.8 (Kopetz [303]) "*A time-constraint is called **hard** if not meeting that constraint could result in a catastrophe*."

    All other time constraints are called **soft time constraints**.

[4] A formal definition of this term is provided in Definition 5.36 on p. 281 of this book.

Many of today's information processing systems are using techniques for speeding-up information processing *on the average*. For example, caches improve the average performance of a system. In other cases, reliable communication is achieved by repeating certain transmissions. On average, such repetitions result in a (hopefully) small loss of performance, even though for a certain message the communication delay can be several orders of magnitude larger than the normal delay. In the context of real-time systems, arguments about the average performance or delay cannot be accepted. "***A guaranteed system response has to be explained without statistical arguments***" [303]. Many modeling techniques in computer science do not model real time. Frequently, time is modeled without any physical units attached to it, which means that no distinction is made between picoseconds and centuries. The resulting problems are very clearly formulated in a statement made by Edward Lee: "*The lack of timing in the core abstraction (of computer science) is a flaw, from the perspective of embedded software*" [330].

– Many embedded systems are **hybrid systems** in the sense that they include analog and digital parts. Analog parts use continuous signal values in continuous time, whereas digital parts use discrete signal values in discrete time. Many physical quantities are represented by a pair, consisting of a real number and a unit. The set of real numbers is uncountable. In the cyber-world, the set of representable values for each number is finite. Hence, **almost all physical quantities can only be approximated in digital computers**. During simulations of physical systems on digital computers, we are typically assuming that this approximation gives us meaningful results. In a paper, Taha considered consequences of the non-availability of real numbers in the cyber-world [522].
– Physical systems can exhibit the so-called Zeno effect. The Zeno effect can be introduced with the help of the bouncing ball example. Suppose that we are dropping a bouncing ball onto the floor from a particular height. After releasing the ball, it will start to fall, being accelerated by the gravitation of the earth. When it hits the floor, it will bounce, i.e., it will start to move in the opposite direction. However, we assume that bouncing will have some damping effect and that the initial speed of the ball after the bounce will be reduced by a factor of $s < 1$, compared to the speed right before the bounce. The case $s < 1$ is also called **inelastic collision**. s is called the restitution. Due to this, the ball will not reach its initial height. Furthermore, the time to reach the floor a second time will be shorter than for the initial case. This process will be repeated, with smaller and smaller intervals between the bounces. However, according to the ideal model of inelastic collisions, this process will go on and on. Figure 1.5 visualizes the height as a function of time (a so-called time/distance diagram) of the inelastic collision.

  Now, let Δ be an arbitrary time interval, anywhere in the time domain. Would there be an upper bound on the number of bounces in this time interval? No, there would not be an upper bound, since bouncing is repeated in shorter and shorter intervals.

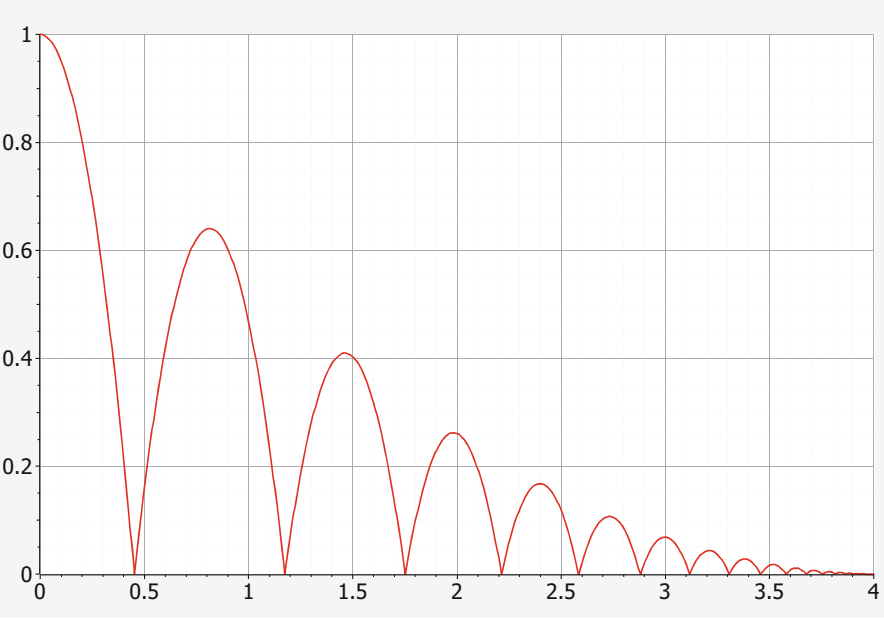

Fig. 1.5 Time/distance diagram of the inelastic collision (© Openmodelica)

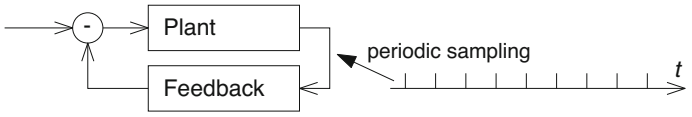

Fig. 1.6 Control loop

This is a special case of the Zeno effect. A system is said to exhibit a Zeno effect, when it is possible to have an unlimited number of events in an interval of finite length [403]. Mathematically speaking, this is feasible since infinite series may be converging to a finite value. In this case, the infinite series of times at which bouncing occurs is converging to a finite instance in time. See the discussion starting on p. 46 for more details. On digital computers, the unlimited number of events can only be approximated.

– Many CPS comprise control loops, like the one shown in Fig. 1.6.

  Control theory was initially based on analog continuous feedback systems. For digital, discrete time feedback, periodic sampling of signals has been the default assumption for decades and it worked reasonably well. However, periodic sampling is possibly not the best approach. We could save resources if we would extend sampling intervals during times of relatively constant signals. This is the idea of **adaptive sampling**. Adaptive sampling is an area of active research [209].

- Traditional sequential programming languages are not the best way to describe concurrent, timed systems.
- Traditionally, the process of verifying whether or not some product is a correct implementation of the specification is generating a Boolean result: either the product is correct or not. However, two physically existing products will never be exactly identical. Hence, we can only check with some level of imprecision whether a product is a correct implementation of the design. This introduces fuzziness and Boolean verification is replaced by fuzzy verification [184, 446].
- Edward Lee pointed out that the combination of a deterministic physical model and a deterministic cyber model will possibly be a non-deterministic model [333]. Non-deterministic sampling can be one reason for this.

Overall, we observe a mismatch between the physical and the cyber-world. Effectively, we are still looking for appropriate models for CPS, but cannot expect to completely eliminate the mismatch.

- Embedded systems must **use resources efficiently**. This requires that we must be aware of the resources needed. The following metrics can be considered in order to evaluate resource efficiency:

1. **Energy:** *Electronic* information and communication technology (ICT) uses electrical energy for information processing and communication. The amount of electrical energy used is frequently called "consumed energy." Strictly speaking, this term is not correct, since the total amount of energy is invariant. Actually, we are **converting** electrical energy into some other form of energy, typically thermal energy. For embedded systems, the availability of **electrical** power and energy (as the integral of power over time) is a deciding factor. This was already observed in a Dutch road mapping effort: "*Power is considered as the most important constraint in embedded systems*" [150].

   Why should we care about the amount of electrical energy converted, i.e., why should there be energy awareness? There are many reasons for this. Most reasons are applicable to most types of systems, but there are exceptions, as shown in Table 1.1.

Table 1.1 Relevance of reasons to be energy-aware

	Relevant during use?		
System type	Plugged	Charged	Unplugged
Example	Factory	Laptop	Sensor network
Global warming	Yes	Hardly	No
Cost of energy	Yes	Hardly	Typically not
Increasing performance	Yes	Yes	Yes
Unplugged uptime	No	Yes	Yes
Problems with cooling, avoiding hot spots	Yes	Yes	Yes
Avoiding high currents, metal migration	Yes	Yes	Yes
Energy a very scarce resource	No	Hardly	Yes

Global warming is of course a very important reason for trying to be energy-aware. However, typically very limited amounts of energy are available to unplugged systems, and, hence, their contribution to global warming is small.[5]

The cost of energy is relevant whenever the amount of energy needed is expensive. For plugged systems, this could happen due to large amounts of consumed energy. For unplugged systems, these amounts are typically small, but there could be cases for which it is very expensive to provide even a small amount.

Increased computing performance usually requires additional energy and, hence, has an impact on the resulting energy consumption.

Thermal effects are becoming more important and have to be considered as well. The reliability of circuits decreases with increasing temperatures. Hence, increased energy consumptions are typically decreasing the reliability. It may be necessary to power-down parts of the system completely to cope with thermal constraints. This effect has been called the **dark silicon** effect (certain areas of silicon chips have to remain unpowered or "dark") [153].

In some cases (like remote sensor nodes), energy is a really scarce resource.

It is interesting to look at those cases where certain reasons to save energy can be considered irrelevant: For systems connected to the power grid, energy is not a really scarce resource. Unplugged systems, due to the limited capacity of batteries, consume very small amounts of energy, and their impact on global warming is small. Systems which are only temporarily connected to the power grid are somewhere between their plugged and unplugged counterparts.

The importance of power and energy efficiency was initially recognized for embedded systems. The focus on these objectives was later taken up for general-purpose computing as well and led to initiatives such as the **green computing initiative** [11].

In general, not only the energy consumption during the **use** of some device is important. Rather, the **fabrication** of the device should be considered as well, due to the energy consumption during fabrication. Hence, we should consider the entire **life cycle** of a product in the form of a so-called life-cycle assessment (LCA) [374]. It is feasible to reduce the impact on the environment by purchasing new devices less frequently.

2. **Run-time:** Embedded systems should exploit the available hardware architecture as much as possible. Inefficient use of execution time (e.g., wasted processor cycles) should be avoided. This implies an optimization of execution times across all levels, from algorithms down to hardware implementations.

[5]This can be demonstrated by means of an example. Consider a mobile phone battery having a capacity of 3600 mAh. We assume an average voltage of 4 V. This results in an energy of 14.4 Wh. A fully charged battery stores as much energy as is consumed by a typical residential gateway (turned on 24/7) in about 1–2.5 h or a TV set in a fraction of an hour.

3. **Code size:** For some embedded systems, code typically has to be stored on the system itself. There may be tight constraints on the storage capacity of the system. This is especially true for **systems on a chip** (SoCs), systems for which all the information processing circuits are included on a single chip. If the instruction memory is to be integrated onto this chip, it should be used very efficiently. For example, there may be medical devices implanted into the human body. Due to size and communication constraints of such devices, code has to be very compact.

   However, the importance of this design goal might change, when dynamically loading code becomes acceptable or when larger memory densities (measured in bits per volume unit) become available. Flash-based memories and new memory technologies will potentially have a large impact.
4. **Weight:** All portable systems must be lightweight. A low weight is frequently an important argument for buying a particular system.
5. **Cost:** For high-volume embedded systems in mass markets, especially in consumer electronics, competitiveness on the market is an extremely crucial issue, and efficient use of hardware components and the software development budget are required. A minimum amount of resources should be used for implementing the required functionality. We should be able to meet requirements using the least amount of hardware resources and energy. In order to reduce the energy consumption, clock frequencies and supply voltages should be as low as possible. Also, only the necessary hardware components should be present, and over-provisioning should be avoided. Components which do not improve the worst case execution time (such as many caches or memory management units) can sometimes be omitted.

Due to resource awareness targets, software designs cannot be done independently of the underlying hardware. Therefore, software and hardware must be taken into account during the design steps. This, however, is difficult, since such integrated approaches are typically not taught at educational institutes. The cooperation between electrical engineering and computer science has not yet reached the required level.

A mapping of specifications to custom hardware would provide the best energy efficiency. However, hardware implementations are very expensive and require long design times. Therefore, hardware designs do not provide the flexibility to change designs as needed. We need to find a good compromise between efficiency and flexibility.

- CPS and IoT systems are frequently collecting huge amounts of data. These large amounts of data have to be stored and they have to be analyzed. Hence, there is a strong link between the problems of **big data** (or machine learning) and CPS/IoT. This is exactly the topic of our collaborative research center SFB 876.[6] SFB 876 focuses on machine learning under resource constraints.

[6] See http://www.sfb876.tu-dortmund.de.

- **Impact beyond technical issues**: Due to the major impact on society, legal, economic, social, human, and environmental impacts must be considered as well:
  - The integration of many components, possibly by different providers, raises serious issues concerning liability. These issues are being discussed, for example, for self-driving cars. Also, ownership issues must be solved. It is unacceptable to have one of the involved companies own all rights.
  - Social issues include the impact of new IT devices on society. This has led to the introduction of the term *Cyber-Physical-Social Systems* (CPSS) [140]. Currently, this impact is frequently only detected long after the technology became available.
  - Human issues comprise user-friendly man-machine interfaces.
  - Contributions to global warming and the production of waste should be at an acceptable level. The same applies to the consumption of resources.
- Real systems are concurrent. Managing **concurrency** is therefore another major challenge.
- Cyber-physical and IoT systems are typically consisting of heterogeneous hardware and software components from various providers and have to operate in a changing environment. The resulting **heterogeneity** poses challenges for the correct cooperation of components. It is not sufficient to consider only software or only hardware design. Design complexity requires adopting a hierarchical approach. Furthermore, real embedded systems consist of many components and we are interested in **compositional design**. This means, we would like to study the impact of combining components [213]. For example, we would like to know whether we could add a GPS system to the sources of information in a car without overloading the communication bus.
- CPS design involves knowledge from many areas. It is difficult to find staff members with a sufficient amount of knowledge in all relevant areas. Even organizing the knowledge transfer between relevant areas is already challenging. Designing a curriculum for a program in CPS design is even more challenging, due to the tight ceilings for the total workload for students [379]. Overall, **tearing down walls between disciplines** and departments or at least lowering them would be required.

A list of challenges is also included in a report on IoT by Sundmaeker et al. [516].

1.4 Common Characteristics

In addition to the challenges listed above, there are more common characteristics of embedded, cyber-physical and IoT systems, independently of the application area.

- CPS and IoT systems use **sensors** and **actuators** to connect the embedded system to the physical environment. For IoT, these components are connected to the Internet.

  Definition 1.9 **Actuators** are devices converting numbers into physical effects.

- Typically, embedded systems are **reactive systems**, which are defined as follows:

  Definition 1.10 (Bergé [567]) "*A reactive system is one that is in continual interaction with its environment and executes at a pace determined by that environment*."

  Reactive systems are modeled as being in a certain state, waiting for an input. For each input, they perform some computation and generate an output and a new state. Hence, automata are good models of such systems. Mathematical functions, describing the problems solved by most algorithms, would be an inappropriate model.
- Embedded systems are **under-represented in teaching and in public discussions**. Real embedded systems are complex. Hence, comprehensive equipment is required for realistically teaching embedded system design. However, teaching CPS design can be appealing, due to the visible impact on the physical behavior.
- These systems are frequently **dedicated toward a certain application**. For example, processors running control software in a car or a train will typically always run that software, and there will be no attempt to run a game or spreadsheet program on the same processor. There are mainly two reasons for this:

  1. Running additional programs would make those systems less dependable.
  2. Running additional programs is only feasible if resources such as memory are unused. No unused resources should be present in an efficient system.

  However, the situation is slowly changing. For example, the AUTOSAR initiative [28] demonstrates more dynamism in the automotive industry.
- Most embedded systems do not use keyboards, mice, and large computer monitors for their user interface. Instead, there is a **dedicated user interface** consisting of push buttons, steering wheels, pedals, etc. Because of this, the user hardly recognizes that information processing is involved. This is consistent with the introduction of the term **disappearing computer**.

Table 1.2 highlights some distinguishing features between the designs of PC-like or data center server-like systems and embedded systems.

Compatibility with traditional instruction sets employed for PCs is less important for embedded systems, since it is typically possible to compile software applications for architectures at hand. Sequential programming languages do not match well with the need to describe concurrent real-time systems, and other ways of modeling applications may be preferred. Several objectives must be considered during the design of embedded/cyber-physical systems. In addition to the average performance, the worst case execution time, energy consumption, weight,

Table 1.2 Distinction between PC-like and embedded system designs

	Embedded	PC-/server-like
Architectures	Frequently heterogeneous very compact	Mostly homogeneous not compact (x86, etc.)
x86 compatibility	Less relevant	Very relevant
Architecture fixed?	Rarely	Yes
Models of computation (MoCs)	C+multiple models (data flow, discrete events, ...)	Mostly von Neumann (C, C++, Java)
Optimization objectives	Multiple (energy, size, ...)	Average performance dominates
Safety-critical?	Possibly	Usually not
Real-time relevant	Frequently	Hardly
Apps. known at design time	Yes, for real-time systems	Only some (e.g., WORD)

reliability, operating temperatures, etc. may have to be optimized. Meeting real-time constraints is very important for CPS but hardly so for PC-like systems. Time constraints can be verified at design time only if all the applications are known at this time. Also, it must be known, which applications should run concurrently. For example, designers must ensure that a GPS application, a phone call, and data transfers can be executed at the same time without losing voice samples. For PC-like systems, knowledge about concurrently running software is almost never available and best effort approaches are typically used.

Why does it make sense to consider all types of embedded systems in one book? It makes sense because information processing in embedded systems has many common characteristics, despite being physically so different.

Actually, not every embedded system will have all the above characteristics. We can define the term "embedded system" also in the following way:

Definition 1.11 Information processing systems meeting **most of the characteristics** listed above are called **embedded systems**.

This definition includes some fuzziness. However, it seems to be neither necessary nor possible to remove this fuzziness.

1.5 Curriculum Integration of Embedded Systems, CPS, and IoT

Unfortunately, embedded systems are hardly covered in the 2013 edition of the Computer Science Curriculum, as published by ACM and the IEEE Computer Society [10]. However, the growing number of applications results in the need for more education in this area. This education should help overcome the limitations of currently available design technologies. Surveys of requirements and approaches to CPS education have been published by the National Academies of Sciences,

Engineering, and Medicine [409] and by Marwedel et al. [379]. There is still a need for better specification languages, models, tools generating implementations from specifications, timing verifiers, system software, real-time operating systems, low-power design techniques, and design techniques for dependable systems. This book should help in teaching the essential issues and should be a stepping stone for starting more research in the area. **Additional information related to the book can be obtained from the following web page:** http://ls12-www.cs.tu-dortmund.de/~marwedel/es-book

This page includes links to slides, videos, simulation tools, error corrections, and other related materials. Videos are directly accessible from: https://www.youtube.com/user/cyphysystems

Users of this material who discover errors or who would like to make comments on how to improve the material should send an e-mail to: **peter.marwedel@tu-dortmund.de**

Due to the availability of this book and of videos, it is feasible and recommended to try out flipped classroom teaching [375]. With this style of teaching, students are requested to watch the videos (or read the book) at home. The presence of the students in the classroom is then used to interactively solve problems. This helps to strengthen problem-solving competences, team work, and social skills. In this way, the availability of the Internet is exploited to improve teaching methods for students actually present at their university. Assignments could use the information in this or in complementary books (e.g., [593], [81], and [174]).

With flipped classroom teaching, existing lab session slots can be completely dedicated to gaining some practical experience with CPS. Toward this end, a course using this textbook should be complemented by an exciting lab, using, for example, small robots, such as Lego Mindstorms™ or micro-controllers (e.g., Raspberry Pie, Arduino, or Odroid). For micro-controller boards which are available on the market, educational material is typically available. Another option is to let students gain some practical experience with finite state machine tools. Teaching from this book should be complemented by a course on machine learning (or data analysis) [188, 204, 453, 560], since the (possibly noisy) values returned by sensors must be interpreted.

1.5.1 Prerequisites

The book assumes a basic understanding in several areas:

- Computer programming (including foundations of software engineering and some experiences with programming of micro-controllers)
- Algorithms (graph algorithms, optimization algorithms, algorithm complexity)
- Computer organization, for example, at the level of the introductory book by J.L. Hennessy and D.A. Patterson [212], including finite state automata
- Fundamentals of operating systems

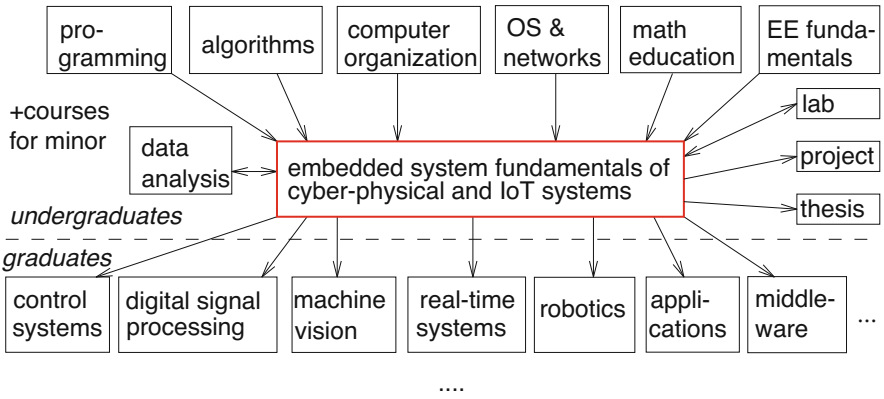

Fig. 1.7 Positioning of the topics of this book

- Fundamentals of computer networks (important for IoT!)
- Fundamental mathematical concepts (tuples, integrals, and linear algebra)
- Electrical networks and fundamental digital circuits such as gates and registers

These prerequisites can be grouped into the courses in the top row of Fig. 1.7.

Missing fundamental knowledge on electrical circuits, operational amplifiers, memory management, and integer linear programming can be compensated by reading appendices of this book. Knowledge in statistics and Fourier transforms are welcome.

1.5.2 Recommended Additional Courses

The book should be complemented by follow-up courses providing a more specialized knowledge in some of the following areas (see the bottom row in Fig. 1.7):[7]

- Control systems
- Digital signal processing
- Machine vision
- Real-time systems, real-time operating systems, and scheduling
- Robotics
- Application areas such as telecommunications, automotive, medical equipment, and smart homes
- Middleware

[7]The partitioning between undergraduate courses and graduate courses may differ between universities.

- Specification languages and models for embedded systems
- Sensors and actuators
- Dependability of computer systems
- Low-power design techniques
- Physical aspects of CPS
- Computer-aided design tools for application-specific hardware
- Formal verification of hardware systems
- Testing of hardware and software systems
- Performance evaluation of computer systems
- Ubiquitous computing
- Advanced communication techniques for IoT
- The Internet of Things (IoT)
- Impact of embedded, CPS, and IoT systems
- Legal aspects of embedded, CPS, and IoT systems

1.6 Design Flows

The design of the considered systems is a rather complex task, which has to be broken down into a number of subtasks to be tractable. These subtasks must be performed one after the other and some of them must be repeated.

The design information flow starts with ideas in people's heads. These ideas should incorporate knowledge about the application area. They must be captured in a design specification. In addition, standard hardware and system software components are typically available and should be reused whenever possible (see Fig. 1.8). In Fig. 1.8 (as well as in other similar diagrams in this book), we are using **boxes with rounded corners for stored information** and **rectangles for transformations on information**. In particular, information is stored in the **design repository**. The repository allows keeping track of design models. In most cases, the repository should provide version management or "revision control," such as CVS [87], SVN [108], or "git" (see https://www.git-scm.com). A good design repository should also come with a design management interface which would also keep track of the applicability of design tools and sequences, all integrated into a comfortable graphical user interface (GUI). The design repository and the GUI can be extended into an **integrated development environment** (IDE), also called **design framework** (see, e.g., [345]). An integrated development environment keeps track of dependencies between tools and design information.

Using the repository, design decisions can be taken in an iterative fashion. At each step, design model information must be retrieved. This information is then considered.

During design iterations, **applications are mapped** to execution platforms, and new (partial) design information is generated. The generation comprises the mapping of operations to concurrent tasks, the mapping of operations to either

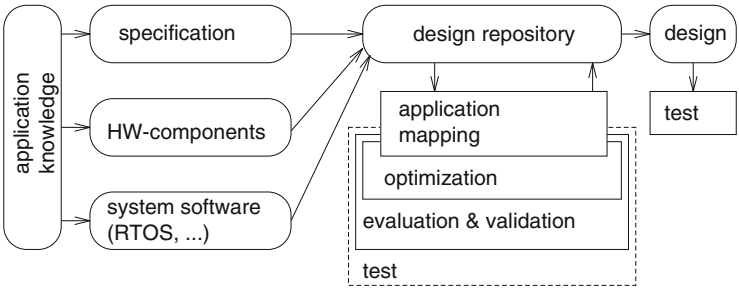

Fig. 1.8 Simplified design information flow

hardware or software (called hardware/software partitioning), compilation, and scheduling.

Designs should be **evaluated** with respect to various objectives including performance, dependability, energy consumption, and thermal behavior. At the current state of the art, usually none of the design steps can be guaranteed to be correct. Therefore, it is also necessary to **validate** the design. Validation consists of checking intermediate or final design descriptions against other descriptions. Thus, each design decision should be evaluated and validated.

Due to the importance of the efficiency of embedded systems, **optimizations** are important. There are a large number of possible optimizations, including high-level transformations (such as advanced loop transformations) and energy-oriented optimizations.

Design iterations could also include **test generation** and an evaluation of the testability. Testing needs to be included in the design iterations if testability issues are already considered during the design steps. In Fig. 1.8, test generation has been included as optional step of design iterations (see the dashed box). If test generation is not included in the iterations, it must be performed after the design has been completed.

At the end of each step, the repository should be updated. Version support would be welcome.

Details of the flow between the repository, application mapping, evaluation, validation, optimization, testability considerations, and storage of design information may vary. These actions may be interleaved in many different ways, depending on the design methodology used. This book presents embedded system design from a broad perspective, and it is not tied toward particular design flows or tools. Therefore, we have not indicated a particular list of design steps. For any particular design environment, we can "unroll" the loop in Fig. 1.8 and attach names to particular design steps.

For example, this leads to the particular case of the SpecC [173] design flow shown in Fig. 1.9. In this case, a particular set of design steps, such as architecture exploration, communication synthesis, and software and hardware compilation are included. The precise meaning of these terms is not relevant in this book. In the case

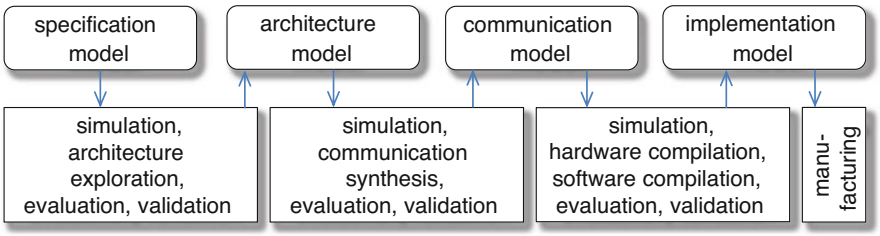

Fig. 1.9 Design flow for SpecC tools (simplified)

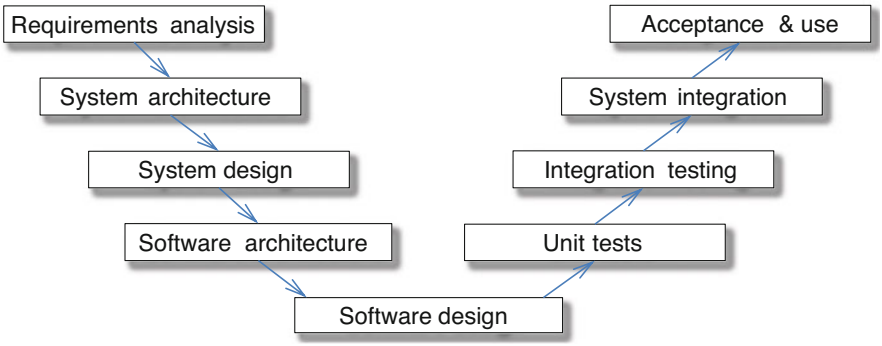

Fig. 1.10 Design flow for the V-model

of Fig. 1.9, validation and evaluation are explicitly shown for each of the steps but are wrapped into one larger box.

A second instance of an unfolded Fig. 1.8 is shown in Fig. 1.10. It is the V-model of design flows [550], which has to be adhered to for many German IT projects.

The model is used especially in the public sector but also beyond. Figure 1.10 very clearly shows the different steps that must be performed. The steps correspond to certain phases during the software development process (the precise meaning is again not relevant in the context of this book). Note that taking design decisions and evaluating and validating designs are lumped into a single box in this diagram. Application knowledge, system software, and system hardware are not explicitly shown. The V-model also includes a model of the integration and testing phase (right "wing") of the diagram. This corresponds to an inclusion of testing into the integration phase. The shown model corresponds to the V-model version "97". The more recent V-model XT allows a more general set of design steps. This change matches very well to our interpretation of design flows in Fig. 1.8. Other iterative approaches include the **waterfall model** and the **spiral model**. More information about software engineering for embedded systems can be found in a book by J. Cooling [109].

Our generic design flow model is also consistent with flow models used in hardware design. For example, Gajski's Y-chart [171] (see Fig. 1.11) is a very

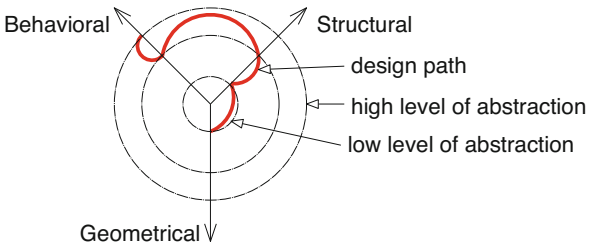

Fig. 1.11 Gajski's Y-chart and design path (red, bold)

popular model. Gajski considers design information in three dimensions: behavior, structure, and layout. The first dimension just reflects the behavior. A high-level model would describe the overall behavior, while finer-grained models would describe the behavior of components. Models at the second dimension include structural information, such as information about hardware components. High-level descriptions in this dimension could correspond to processors and low-level descriptions to transistors. The third dimension represents geometrical layout information of chips. Design paths will typically start with a coarse-grained behavioral description and finish with a fine-grained geometrical description. Along this path, each step corresponds to one iteration of our generic design flow model. In the example of Fig. 1.11, an initial refinement is done in the behavioral domain. The second design step maps the behavior to structural elements and so on. Finally, a detailed geometrical description of the chip layout is obtained.

The previous three diagrams demonstrate that a number of design flows are using the iterative flow of Fig. 1.8. The nature of the iterations in Fig. 1.8 can be a source of discussions. Ideally, we would like to describe the properties of our system and then let some smart tool do the rest. Automatic generation of design details is called **synthesis**.

Definition 1.12 (Marwedel [370]) "***Synthesis*** *is the process of generating the description of a system in terms of related lower-level components from some high-level description of the expected behavior.*"

Automatic synthesis is assumed to perform this process automatically. Automatic synthesis, if successful, avoids many manual design steps. The goal of using automatic synthesis for the design of systems has been considered in the "describe-and-synthesize" paradigm by Gajski [172]. This paradigm is in contrast to the more traditional "specify-explore-refine" approach, also known as "design-and-simulate" approach. The second term stresses the fact that manual design typically has to be combined with simulation, for example, for catching design errors. In the traditional approach, simulation is more important than in automatic synthesis.

1.7 Structure of This Book

Consistent with the design information flow shown above, this book is structured as follows: Chapter 2 provides an overview of specification techniques, languages, and models. Key hardware components of embedded systems and the cyphy-interface are presented in Chap. 3. Chapter 4 deals with system software components, particularly embedded operating systems. Chapter 5 contains the essentials of embedded system design evaluation and verification. Mapping applications to execution platforms is one of the key steps in the design process of embedded systems. Standard techniques (including scheduling) for achieving such mapping are listed in Chap. 6. Due to the need for generating efficient designs, many optimization techniques are needed. From among the abundant set of available optimization techniques, several groups are mentioned in Chap. 7. Chapter 8 contains a brief introduction to testing mixed hardware/software systems. The Appendix comprises prerequisites for understanding the book, and it can be skipped by students familiar with the topics covered there.

It may be necessary to design special-purpose hardware or to optimize processor architectures for a given application. However, hardware design is not covered in this book. Coussy and Morawiec [113] provide an overview of high-level hardware synthesis techniques.

The content of this book is different from the content of most other books on embedded systems or CPS design. Traditionally, the focus of many such books is on explaining the use of micro-controllers, including their memory, I/O, and interrupt structure. There are many such books [38, 175–177, 279, 317, 425]. We believe that, due to the increasing complexity of embedded and cyber-physical systems, this focus has to be extended to include at least different specification paradigms, fundamentals of hardware building blocks, the mapping of applications to execution platforms, as well as evaluation, validation, and optimization techniques. In the current book, we will be covering all these areas. The goal is to provide students with an introduction to embedded systems and CPS, enabling students to put the different areas into perspective.

For further details, we recommend a number of sources (some of which have also been used in preparing this book):

- Symposia dedicated toward embedded/cyber-physical systems include the Embedded Systems Week (see http://www.esweek.org) and the Cyber-Physical Systems Week (see http://www.cpsweek.org).
- The web site of the virtual CPS Organization in the USA contains numerous links to current projects and their results [115].
- The web page of a special interest group of ACM [9] focuses on embedded systems.
- The web site of the European network of excellence on embedded and real-time systems [25] also provides numerous links for the area.
- A book written by Edward Lee et al. also includes physical aspects of cyber-physical systems [335].

- Approaches for embedded system education are covered in the Workshops on Embedded Systems Education (WESE; see [89] for results from the workshop held in 2018) and in proceedings of the first (and only) Workshop on CPS Education [424].
- Other sources of information about embedded systems include books by Laplante [322], Vahid [552], the ARTIST road map [63], the "Embedded Systems Handbook" [614], and books by Gajski et al. [174], and Popovici et al. [457].
- There are a large number of sources of information on specification languages. These include earlier books by Young [609], Burns and Wellings [80], Bergé [567], and de Micheli [124]. There are a huge amount of information on languages such as SystemC [407], SpecC [173], and Java [71, 131, 574].
- Real-time scheduling is covered comprehensively in the books by Buttazzo [81], by Krishna and Shin [310], and by Baruah et al. [41].
- Approaches for designing and using real-time operating systems (RTOSes) are presented in a book by Kopetz [303].
- Robotics is an area that is closely linked to embedded and cyber-physical systems. We recommend the book by Siciliano et al. [487] for information on robotics.
- There are specialized books and articles on the Internet of Things [185, 192, 193].
- Languages and verification are covered in a book by Haubelt and Teich (in German) [206].

1.8 Problems

We suggest solving the following problems either at home or during a flipped classroom session [375].

1.1 Please list possible definitions of the term "embedded system"!

1.2 How would you define the term "cyber-physical system (CPS)"? Do you see any difference between the terms "embedded systems" and "cyber-physical systems"?

1.3 What is the "Internet of Things" (IoT)?

1.4 What is the goal of "Industry 4.0"?

1.5 In which way does this book cover CPS and IoT design?

1.6 In which application areas do you see opportunities for CPS and IoT systems? Where do you expect major changes caused by information technology?

1.7 Use the sources available to you to demonstrate the importance of embedded systems!

1.8 Which challenges must be overcome in order to fully take advantage of the opportunities?

1.9 What is a hard timing constraint? What is a soft timing constraint?

1.10 What is the "Zeno effect"?

1.11 What is adaptive sampling?

1.12 Which objectives must be considered during the design of embedded and cyber-physical systems?

1.13 Why are we interested in energy-aware computing?

1.14 What are the main differences between PC-based applications and embedded/CPS applications?

1.15 What is a reactive system?

1.16 On which web sites do you find companion material for this book?

1.17 Compare the curriculum of your educational program with the description of the curriculum in this introduction. Which prerequisites are missing in your program? Which advanced courses are available?

1.18 What is flipped classroom teaching?

1.19 How could we model design flows?

1.20 What is the "V-model"?

1.21 How could we define the term "synthesis"?