DLR

OFFIS
INSTITUT FÜR INFORMATIK

CARL VON OSSIETZKY universität OLDENBURG
FAKULTÄT II
INFORMATIK, WIRTSCHAFTS- UND RECHTSWISSENSCHAFTEN

# System-Level Design

2.01.334, Summer 2024

## Exercise 1

**Methodologies, Models, Languages**

**Assigned:** April 8, 2024
**Due:** April 15, 2024

**General Instructions**

1. Please submit your solutions via mail to your advisors (mail addresses can be found at the end of this document). Submissions should include a single typewritten PDF file with the writeup and a single Zip or Tar archive for any supplementary files (e.g. source files, which has to be compilable by simply running `make` and should include a README file with instructions for running each model).

2. You may discuss the problems with your classmates but make sure to submit your own independent and individual solutions. You are allowed to work in groups with up to three members. In this case, please put the full names of all three group members on the cover page of you submitted solutions.

3. Some questions might not have a clearly correct or wrong answer. In general, try to write down your arguments and reasoning how you have arrived at your solutions.

---

**Task 1: System Design**

During design space exploration as part of the system design process, the target system architecture and its key architectural parameters are decided on. These design decisions have a major influence on the final design quality metrics such as (i) performance, (ii) power, (iii) cost, and (iv) time-to-market:

**a)** Briefly discuss how the following target platform styles rate in relation to each other in terms of the metrics listed above:

- A pure software solution on a general-purpose processor
- A general-purpose processor assisted by a custom hardware accelerator/co-processor
- A general-purpose processor and a specialized processor (DSP or ASIP)

*This task cannot be answered unambiguously because DSPs are sometimes also referred to as coprocessors. Furthermore, it is not clear what scope the custom hardware accelerator has. This can include functions from a small driver module to a more complex function (e.g. a discrete cosine transformation) implemented in hardware.*

| *component* | *performance* | *power* | *cost* | *time to market* |
|---|---|---|---|---|
| *Pure SW on GPP* | *bad* | *poor* | *low cost, because very common* | *fast, because available* |
| *GPP + custom HW/coprocessor* | *better depending on design of HW-Acc* | *also depending on HW-Acc (if most of the function is shifted to highly power optimized HW-Acc power consumption can be very low)* | *HW-Acc can be bought ready (small increase) or developed by the customer (very high costs)* | *if HW-Acc is developed, very long* |
| *GPP + DSP/ASIP* | *very good performance; GPP is relieved* | *depending on DSP and ASIP savings are possible* | *more expensive than first solution; if ASIP is developed even more expensive* | *moderate in case of available DSP/ASIP, very high in case if DSP/ASIP development* |

**b)** Try to sketch a potential simple strategy for exploring the design space for a given application under a given set of constraints/requirements.

*If the solution is to be finished quickly and the focus is not on energy consumption or performance, then a solution with a GPP is to be preferred. This is cheap in large quantities and can be developed quickly. To improve the performance or the energy consumption a little bit, smaller (related to the functional range) coprocessors or HW accelerators can be used. There is a wide spectrum of ready-made solutions. If a HW-Acc is to be developed, then this becomes an expensive and time-consuming solution. For tasks from the digital signal processing DSPs fit very well. These are also available in different variants and can be adapted or programmed for the respective task.*
*The sketched simple iterative approach is: starting with an initial pure SW on a GPP solution and further adding more customized HW as required. Anyhow. without a systematic System-Level Design approach it can take a long time and it is not guaranteed that an acceptable final solution can be found in time (or even at all).*

**Task 2: Languages**

**a)** Why are sequential programming languages (e.g. C/C++/Java) considered to be insufficient for embedded system specification and design?

*These languages do not support concepts such as concurrency, interprocess communication, or provide sufficient concepts to model temporal behavior.*

**b)** Why are hardware design languages (e.g. VHDL/Verilog) considered to be insufficient for embedded system specification and design?

*These languages are basically designed for the description of structural hardware modules consisting of parallel processes that communicate through signals (event or value changes).*

*HDLs are unsuitable for the description of an entire system since they lack appropriate support for the description of the overall system on an implementation independent level. Moreover, SW can not be appropriately represented, mainly due to the lack of a more expressible type system.*

**Task 3: Language Concepts**
We have discussed the concepts of synthesizability, orthogonality and separation of concerns in languages.

**a)** What are the requirements for languages to be synthesizable?

*A language that is synthesizable if it allows an automatic transformation from a behavior into a structural representation which is execution semantics preserving (i.e. the behavior of the system is not changed). This in return requires that the language has a clean execution semantics (see "Model of Computation" lecture). Moreover, a system-level design language shall allow the automatic transformation of an implementation independent behavior description into a HW and SW implementation. Furthermore, it shall support the reuse of IP (Intellectual Property) components. These are components that have already been designed in previous project and support reuse in a new project.*

**b)** Briefly define what orthogonality is? What is separation of concerns?

*Orthogonality describes the free combinability of components, whereby these must not negatively influence each other. Examples for this are hierarchy and modularization. Seperation of Concerns is understood in this context as the separation of computation and communication. Orthogonality implies separation of concerns.*

**c)** Try to show an example other than the ones discussed in class of non-orthogonal concepts or constructs in a language.

*Arrays in the C language are an example of this. These cannot be used as the return value of a function, since the complete array cannot be returned but just a pointer/reference to it.*

**d)** Name two concerns that should be separated as they cover orthogonal aspects? Show a short code excerpt that demonstrates a non separated code and code that separates these concerns (in C/C++/SpecC/SystemC or similar).

1. *data types and operations on it*
2. *exception handling and regular functional code in e.g. C++*

**Task 4: Specification**
Writing a good specification model is essential to a successful design process. One important characteristic of a specification model is to be free of implementation detail. Please briefly outline why this is important with respect to the top down design flow.

*On such a high abstraction level no statement can be made about implementation details. Depending on the design approach, the choice of platform is made very late in order to have sufficient degrees of freedom during the exploration and development process. Wrong decisions in early phases (such as early (assumed) implementation target specific optimizations) can result in high costs if these optimization have to be removed again later in the design process.*

**Task 5: Design Methodology**

**a)** Compare and contrast a top-down, bottom-up and meet-in-the-middle design methodology.

| *design methodology* | *description* |
|---|---|
| *top down* | *platform unknown; standard approach; only one layout; system metrics not known before synthesis.* |
| *bottom up* | *starting from platform; libraries are created on each level; libraries very extensive; because requirements of higher abstraction levels are not known; application is composed of small modules; individual modules can be optimized (in former times without CAD tools)* |
| *meet in the middle* | *combination of both approaches; many low level tools available (libraries, IPs)* |

**b)** What type of methodology is a platform-based approach? Describe a platform-based methodology and flow.

*In the platform-based approach, the system is started from a ready-made platform. All components of this platform are described in the manufacturer's libraries and own components can be added to the overall system. Many off-the-shelf Systems-on-a-Chip have existing processor cores, peripherals, memories and bus-systems where no or only little HW customization is possible. The system under design has to be implemented on the existing fixed HW elements in the System-on-a-Chip.*

**c)** Why do we perform computation design before communication design in the SpecC/SCE methodology?

*The communication requirements can only follow to the structural choice of the computation elements (e.g. how many processing (GPP and/or custom HW) and memory elements are being used and which communication interfaces do they have?). Specific communication elements like buses, cross-bar-switches or Network-on-Chip (NoC) with their respective protocols can then be placed to physically connect the processing and memory elements. The communication refinement then further refines the implementation of the synchronisation and communication of the specification model on these physical communication elements. The other way around: Starting with a physical communication network and attaching the processing and memory elements to it would be possible, but the communication of the specification model has still to be mapped/implemented on the physical communication network and this approach limits the capabilities to explore different design solutions in order to find the most suitable solution for the final implementation/product.*

---

**Administrative**

Advisors: Kim Grüttner <kim.gruettner@dlr.de>
Jörg Walter <joerg.walter@offis.de>
Henning Schlender <henning.schlender@dlr.de>
Sven Mehlhop <sven.mehlhop@offis.de>