IL2212 EMBEDDED SYSTEMS

Reflection Assignment for Seminar 1

Weikai Zhou

February 8, 2022

Reflection Assignments for Seminar 1

1. **Models are important in the design process. How can models support the design process at different levels of abstraction? What are suitable models in a design process for safety-critical multiprocessor embedded software systems and how should the models be used? Please also relate your answer to Figure 1.3 and 1.5 in the lecture notes.**

   Figure 1.3 shows the how the abstraction gap is bridged between an abstract high-level specification and a detailed low-level implement. At initial steps, the model needs to be modelled at a high level of abstraction, which gives people a rough understanding of the whole system and pave the way for the future design. Then, people try to partition the high-level abstracted model, which is separated into different subsystems. These subsystems are then be abstracted and expressed by the model again. They can be analysed and designed separately. During this process, with the help of the model, people can refine the system easily. The level of abstraction can be decreased now since people have a deeper understanding and implementation of the system. It can be concluded that during the design process, the initial specification model is converted into more detailed models using refinement steps, which yield a model at a lower level of abstraction.This process is repeated again and again until the whole system is built and implemented successfully, and different sub-models are built to help people understand, design and implement the corresponding system.

   I believe a large amount of models are suitable for safety-critical multiprocessor embedded software systems as long as they can become a good system model. They should

   - define the functionality the system has to perform,
   - enable to simulate or formally verify the functionality of the system, so that designers can confirm that the intended functionality is modelled correctly,
   - use a formal language with a standardised semantics, so that the model has one clear meaning,
   - abstract from low-level details, but provides at the same time a path to the final implementation,
   - abstract from implementation techniques, so that the final implementation can use different implementation techniques, like hardware, bare-metal software or real-time operating systems, and
   - allow the designer to focus on the most important aspects. (According to Figure 1.5, designer should focuses on design constraints, platform architecture, mapping and schedules and so on for modeling a safety-critical multiprocessor embedded software system. Therefore, they should, for example, be easily scheduled for designers to analyze their behavior during several periods. They should also be able to reflect the constraints of the design including the ability to be real-time and so on. They should be helpful to analyze the performance of the system.)

And we should use different models based on the functionality, requirements, level of abstraction and other factors. Using UML as an example, which is one of the two dominating languages for the modelling of software systems, there are different models and each of them focuses on different aspects of the system. E.g., a sequence diagram is especially suitable for describing the interactions between two parts of a system; a class diagram is suitable for describing the static structure of the system. Therefore, we should choose correct models to model the system. For example, during the partitioning of the system, one can use the class diagram to represent the relation between different systems and sub-systems. One can use the sequence diagram to represent the actors, data flows, and computation sequences of the system. Hence, each model has its own functionality and advantages, and we should be able to maximize their strengths to model the system.

2. **Explain the term *model of computation* in your own words. What do you view as the largest benefits of the *theory on models of computation*? How can these benefits be exploited in a design process for safety-critical multiprocessor embedded systems? Would you prefer *analysable* or *expressive* models in the design process?**

   Model of computation describes the computation process of the output given the input, which is the constitution of the system. It shows the connection between the processes or actors by signals, which can be regarded as the communication mechanism of the system. It also shows the concurrency of the system.

   The largest benefits are that MoCs are formally defined and they can be mathematically analyzed and predicted, which enables the development of tools for simulation, performance analysis and design space exploration, formal verification and synthesis and code generation. That is to say it is a formally analyzable and predictable model.

   For the analysability, if we know the consume rate and produce rate of each actor, for example, we can get a Periodic Admissible Sequential Schedule (PASS), if it is a schedule for a single processor computer, or Periodic Admissible Parallel Schedule (PAPS), if it is a schedule for a parallel computer. Besides, we can also analyze other important aspects of the system if enough information is given. For the predictability, since we can know the schedule of the system, we may be able to predict the worst case execution time, possible execution time, even performance and so on. With the analysability and predictability, we can know the system in a more theoretical way before real implementation, which can save people a lot of time and money.

   There is a trade-off between expressiveness and analysability. A high expressiveness means that the modelling paradigm enables to express many different models or programs, while a high analysability means that the modelling paradigm enables to analyse the model and to prove or derive certain properties of the model. Personally speaking, I prefer the analysability model because it enables us to check the design constraints, predict the schedules, measure the performance metrics and so on during the design process, giving us an insight into the system. However, the system is increasingly complex nowadays and the requirement for expressiveness is high. Therefore, expressiveness is quite important. We should try to increase the level of expressiveness without sacrificing too much analysability.

3. **The course uses ForSyDe as modelling language. Compare the ForSyDe modelling language with other modelling/programming languages that you have used before. What are the advantages and disadvantages of the ForSyDe modelling language? How should the ForSyDe modelling language be used in a system design process? Is there a need for other languages in the design process and how would they interact with the ForSyDe modelling language?**

   Advantages: It provides several libraries for different MoCs. It is a pure mathematical function from input signals to output signals. It specifies the consume rate and produce rate of the actor. And it can also be changed during the execution time. Therefore, it reveals part of the inner process of the execution and provides flexibility for us. However, for other languages, it is usually not specified and the consume rate is also can be hardly changed. Since the input variables of the function are generally not in the parenthesises, they are not required to input into the function at the same time.

For example, *function input1 input2*, we can first use *temp = function input1* and then use *temp input2* to get the correct answer. While in other, we are generally not allowed to do so.

Disadvantages: It seems that a few people use this language and it can be hard to find others who encountered similar problems online and study their solutions to these problems. Therefore, in most cases, we can only read the documentation and lecture notes and try to solve the problem by ourselves. While for other modelling/programming languages like c/c++, a lot of people are using them and we can easily find the solutions to some problems in online communities. The function name and its input variables are just simply put in one row without any parenthesises, and in some cases this expression can lead to confusion. While in other languages, the variables are usually in the parenthesises, meaning that the structure of the code is much clearer and is easier to understand.

In a system design process, we usually start at a high level of abstraction and focus on functionality rather than go directly into the detailed low-level implementation. People should use the ForSyDe to model and implement the signals, processes and systems. And people should also use the ForSyDe to analyze, predict and verify whether the design has achieved the goals. However, using only ForSyDe is not enough since it is usually used for abstraction and model the system. When we really go down to the detailed implementation, some other languages may be used like c/c++, VHDL and so on. Based on the hardware platform we are using, we may choose different languages. And with the help of ForSyDe, we have a basic model of the system and we can use other languages to transform the model into a more detailed one with some other small functions.