IL2212 EMBEDDED SOFTWARE

Reflection Assignment 1

Henrik Borg

January 27, 2019

1 Part A - Embedded Software Design

The topic of the this part of the reflection assignment are the special characteristics of embedded software and their impact on the design flow.

1.1 What are the special characteristics of embedded software development?

The system have to behave correct. Often hard to update after system has been deployed. Sometimes even directly after production, due to “deep built into” the product, no external interface for reprogramming. Therefore it is really important to get the system to work according to specifications before shipment of product.

The system must be highly predictable, especially if there are hard requirements.

It can be hard, or at least difficult, to extract enough and correct information regarding the environment of the implementation. Communication protocols, functions/user cases, timely requirements and so on.

1.2 What are the main problems and challenges for embedded software design?

Often part of a critical or semi critical systems, and therefore must perform in a timely manner. Missing one deadline can be catastrophic.

Even if the system is not critical, at all. The feeling for the customer using the product is that it just has to work. A little lagging of 100 ms to 500 ms is often okay for the end customer, but seldom more. Take a second and think how it would feel if it would take 10 s for the TV to react to pressed buttons on its remote control. In most cases the customer expect quite high responsiveness. If not responsive enough, you can loose marked shares.

The system must often work “forever”. Meaning that no errors are allowed during the time it is activated.

Some “nice features” in the processor can give big enough variations in time execution and by that making it harder to predict the timely behavior of the system. Such things can be instruction execution pipelines, caches, DMA, change in the ALU between different versions of the processor.

1.3 What do you view as possible solutions to tackle these problems?

We can use real-time theory to make sure that our system fulfill the timely requirements.

For the customer’s feeling of the responsiveness of the system we should use real-time theory during the SW development. Something the HW team also must think about.

Develop functions that has few possible execution paths. This will reduce the timing anomalies.

During development of functions we must check that they have a high functional safety. We might not be able to update them after production/deployment. Using unit testing or some other SW testing method. Testing in special test beds and real life testing is imposed.

Select a processor with not too many “nice features” and the timing anomalies will be smaller.

2 Part B - Real-Time Theory

The topic of this part of the reflection assignment is the real-time theory based on periodic, aperiodic and sporadic tasks.

2.1 What do you view as the largest achievements of the real-time theory?

The development of simulations (even handmade graphs) and algorithms for offline calculations of the timely behavior of the system in order to make sure the system behaves according to the timely specifications. To do that with sporadic and aperiodic tasks that interfere with the periodic tasks is quite of an achievement.

The algorithms for online calculations can be seen as secondary achievements compared to the offline algorithms. Might sound strange because online algorithms is probably harder to develop. If we cannot offline guarantee the timely behavior of the system, there is little use of implementing online algorithms. Therefore the online algorithms can be seen as lesser of an achievement compared to the offline algorithms.

Use of Petri-Nets and precedence graphs to check if a deadlock can occur. These graphs can be extended for different execution paths, not just between jobs, but also inside the functions.

2.2 How can the real-time theory be used in an industrial design flow?

Real-time theory should always be used in the industrial design flow. Even if it quite soon might show that it is not needed. Just to make sure it is not forgotten, or realized in a very late state that it is needed.

By using the real-time theory one can easily, in often an early state, figure out what kind of constraints is needed to make sure the system will behave as intended and needed. Are there any precedence constraints? Probably yes. Are there any timing constraints? Most probably yes, even if they are not hard, or even firm. Some timing constraints are more for convenience for the user, so they do not become stressed or irritated by waiting unnecessarily long. Are there any resource constraints? Probably yes. At least memory wise. Even if the system only has on task, and that task need a resource, it is a resource constraint. Even if something as simple as a button, or an LED might not always be seen as a necessary resource from the SW development point of view, they are needed and a necessary resource from a holistic point of view. I guess it here is shown that I come from the HW side of thinking.

Parts of industrial plants are designed and build with a real-time view of thinking. As an example we can take the assembly line for cars. For better economy for the plant, and cheaper cars to the customers, no assembly station should be waiting for tokens to work with from the previous station.

2.3 What do you view as the biggest limitations of the real-time theory?

The fact that the functions have been more and more complex over time and by that often gives more and more possible execution paths through the functions makes it harder and harder to come up with new algorithms, or to extend exiting ones to take account for these variations in execution times.

Another big problem is the more and more communications between different systems. It is often quite hard to make good estimations of the timings of these communications, they are often stochastic. The differences in these timings makes it harder to take into account the length of self suspensions or the length between two jobs with precedence constrains.

While more and more functions are squeezed together to execute on fewer and fewer processors makes the using of real-time theory more important, and also more harder.

2.4 Give suggestion how to overcome the limitations mentioned in the previous question!

Try to make the functions simpler/less complex, with few possible execution paths. Then the timely behavior will be easier to predict.

With the use of probability theory we can at least get some estimations of the time lengths of communication (sending, waiting for response, receiving) with other systems. If the processor running the time critical jobs does not need to take into account for retransmissions, “garbage” coming as input from the communication line, the timely behavior of system of that processor will be much easier to predict. Can be achieved by adding another processor to do that work, or use the processor that might already exist in the communication module.

Sometimes it can be better to add an extra processor that only runs the utmost critical parts of the system. By doing that, you get the rest of the system running on the processor with the many less critical, or non-critical jobs less time constrained.