IL2206 Embedded Systems

Reflection Assignment 1

Weikai Zhou

1 Reflection Assignments for Seminar 1

1.1 Give three new examples (not mentioned previously in the lectures or lecture notes) for typical safety-critical embedded systems from different application domains.

1.1.1 Automobile Gearbox

Automobile gearbox is one of the examples of safety-critical embedded systems. Nowadays, many cars are equipped with an automatic gearbox, which can shift gears according to the speed of the car automatically so that it can save people from the trouble of shifting gears manually and decrease the possibility of shut down of the car during the journey. However, if there are some issues with the gearbox, it will lead to incorrect gear changes, or even complete loss of power, which is a horrible circumstance especially on the expressway because the vehicles behind may not be able to brake quickly enough so that a rear-end collision can happen and people's lives are in danger.

Volkswagen Group once had problems with its DSG gearbox. Some customers experienced the sudden failure of the gearbox and several car accidents happened. The issue had been exposed on the Internet and had affected the reputation of Volkswagen Group to some extent. Volkswagen Group had no choice but to extend the warranty, update the software of its DSG gearbox, and even recall some car models.

1.1.2 Nuclear Reaction Control System

Nuclear reaction control system is another example of safety-critical embedded systems. It is widely used in nuclear power plants and some other fields related with nuclear reactions. It can keep the speed of nuclear reaction within a reasonable range by changing the density of boric acid and neutron. Most importantly, it ensures the power of the reactor doesn't exceed the ability to take heat away, otherwise the reactor could overheat and burn down. Besides, when natural disasters or other factors that can lead to the destruction of nuclear power plants happen, the system should be able to take actions immediately to stop the reaction and cool down the nuclear reactor to prevent nuclear leakage.

There was a severe nuclear leakage at Fukushima, Japan. When the earthquake happened, the nuclear reaction was stopped successfully by the system, and the emergency generator was activated to supply power. However, the design of the system was not perfect and it could not function properly when a tsunami came. Therefore, the tsunami coming after the earthquake caused station blackout, and emergency generator also failed. The water pump stopped working and the reactor was overheat and burned through the reaction vessel, leading to severe leakage and environmental disruption.

1.1.3 Elevator Control System

Elevator control system is also an example of safety-critical embedded systems. Almost every elevator has such a system to guarantee safe operation, no matter it is a lift or an escalator. For a lift, the system is generally to control the correct going up and down of the lift, opening and close of the door, and emergency braking in certain situations. For an escalator, the system is generally to detect whether there is anything stuck in the escalator and stop the escalator timely. And the system needs to be maintained regularly to make sure it works properly. If there are some problems or failures in the system, the elevator cannot function properly or stop timely in emergency situations. In extreme cases, people could die for this. For example, a lift may drop rapidly and cannot be stopped, and an escalator may stuck people and continue to operate to roll people into the treads.

1.2 What do you consider the main problems and challenges in the design process of safety-critical embedded systems?

In my opinion, the main problems and challenges may be that how to build a hard real-time system. As mentioned in the lecture slides, "the system has to react to the environment at the right time instance, otherwise there can be fatal consequences". To build a real-time system, it requires a better harmony in both hardware and software. For example, how to detect and handle interrupts needs the sensors or other hardware devices to input the signals. It also needs the processor to process such signals in a short time. And the software should activate the interrupt mechanism to handle it. Through this process, many different aspects have participated in it. And to arrange them in a perfect manner so that the system becomes a hard real-time system is really difficult and time-consuming.

Even if the system has been implemented, it still needs to be verified that it meets the requirements. However, it is difficult to estimate the real-time performance since the average time and worst case time can differ from each other hugely. Besides, the test should be as varied as possible so that it can try to mimic real usage and find the performance in worst cases. Therefore, the cost for test can go up easily.

1.3 What do you consider the most relevant properties for an embedded computing platform designed for safety-critical systems. Motivate! What does this mean for the characteristics of the individual components (processor, memory system, interconnection network, peripherals) that build the platform for a safety-critical embedded systems?

As mentioned above, real-time is one of the most important features of safety-critical systems. Therefore, the most relevant properties should be that the devices can execute their functions in time. In order to achieve this,

For processor: First, there can be both general purpose processors and specialized purpose processors, which can reduce the time for certain functions. Also, the processors need to be properly pipelined or even add more stages so that the execution time for each stage is short enough to finish the execution in time.

For memory system: The multi-level cache can be used to increase the hit rate and decrease the access time. Depending on the task and the code, temporal locality and spatial locality should be chosen to increase performance. For example, temporal locality is usually used for instructions in a loop, and spatial locality is usually used for sequential instructions and array data. Also, we should choose correct associative method for the cache based on the requirements. For example, 1-way associative is low in hit rate but low in access time, while fully-associative is high in hit rate and also high in access time.

For interconnection network: Both synchronous and asynchronous bus have their own advantages and disadvantages. For example, it is easier to design a synchronous bus since it requires little extra logic. But all the devices share the same clock signal so that the clock signal generally should be at the pace of the slowest device. Therefore, the performance of synchronous bus may not be the best. While asynchronous bus can solve this problem, it also requires extra work to design the system. In my opinion, we may use different clock signals at different paces to implement the whole system so that every device can maximize its performance. Slave-side arbitration can also be considered when building a high-performance system. But the hardware cost can increase dramatically.

For peripherals: Buffers are important for peripherals, where the important information can be stored in buffers. Therefore, buffers can decouple activities and the system can do other tasks when waiting for the receiver or sender (polling). Since buffers are storage units, the size of them should be carefully chosen. If the size is small, there may be overflow. And if the size is large, the cost will increase.

1.4 Safety-critical systems execute a lot of software. How can a time-predictable behaviour of the software be supported by the hardware?

I think the general concept is to make things deterministic, which means the time of each step is generally known. Then the software can predict the time based on the processes or steps about to experience.

Therefore, a timer may be needed. And the design of hardware should be synchronized. Then, a deadline can be given to some operations so that they should be finished in several clock circles. Then, based on the number of clock circles, the time can be estimated. For those operations that fail to finish within required clock circles, a worst case timing should be assumed.

The latency of access memory should be designed almost the same so that the time needed for access memory is known. And for multiple memory access requests, the arbitrate should be designed to arrange them in order so that the time can also be predicted.

And the hardware should be designed in a way that different tasks will not affect each other. Otherwise, the time needed will change when it runs together with different software A and software B.