Complexities of embedded systems

Online: www.clarinox.com

Printed RadioComms Asia Pacific January 2011 issue
http://www.radiocomms.com.au/articles/44955-Complexities-of-embedded-systems

Phone: 03 9095 8088

Increasing complexity has long been a major embedded systems developer concern. Some industries have seen this trend earlier than others but it appears none will escape the trend. It is just a matter of time until this is a concern across every market segment.

This trend has certainly been true for radio communications equipment companies as they deal with multiple wireless, increased protocol complexity, increased user interface demands, increasing audio quality requirements, increased security and so forth.

Attempts to deal with this increasing complexity have come from both hardware and software angles, however, there is a shift from hardware-driven designs to software-driven designs.

Hardware is, of course, important and must be used for cases where software cannot do the job, such as for extreme performance, when very precise timing is needed, or when an interface to analog signals is required; however, the proportion of embedded project spend is moving away from majority hardware towards majority software.

An example here is that the medium access control (MAC) layer of most communication protocols (that is responsible for establishing connections, paging remote devices, encoding and decoding data) is now usually implemented in software, despite the fact that these protocols usually require precise timing information.

On the hardware side, increased capability for reasonable price is driving up the uptake of hardware components such as the use of multiple processors.

On the software side, complexity has been dealt with through an increasing percentage of projects using operating systems and middleware. While applications that run directly on embedded CPUs are still around, the total embedded market is being eroded with the use of various embedded operating systems.

The increases in hardware capability for reasonable cost have resulted in, even, operating system cores that were originally designed for PCs can now be used in embedded environments with little modification (eg, Linux).

There is a variety of communication interfaces and buses (for communication between the embedded CPU and peripherals) being used in the market.

Part of this complexity may be due to a lack of unifying standards. This is a reflection of the complex nature of embedded systems and applications, which calls for a

diversity of tools and techniques to be employed during the embedded system design life cycle.

An OS might be a perfect fit for one application but utterly useless for another. This has traditionally led to embedded code being written and optimised for a specific operating system, microprocessor and embedded environment.

With this approach, only a tiny portion of code can be re-used without major modifications in consecutive projects, which drives up project costs.

To circumvent this intrinsic complexity of embedded designs, some engineers have started using virtual embedded environments. These provide programming and debugging tools that are designed to be independent from the actual physical system at which they are aimed at being mapped.

The advantage is removal of the need to know the actual physical system. The disadvantage is a more powerful embedded processor, increasing cost, and one that software cannot exploit features which are exclusively designed for the underlying physical system.

This extra cost may be justified by less design complexity and shorter design life cycle; however this needs to be considered on a project to project basis.

An alternative approach, which has been adopted by many companies, is to base the software on middleware, according to ‘Forecast 2010: What Is in Store for Embedded Developers’.

The use of ‘roll your own’ [embedded communications] middleware, it says, consumes more than 50% of the market - notwithstanding the lack of scalability, the large cost of managing and supporting deployed systems and the unnecessary complexity of such systems. Of high relevance to the radio industry is the use of embedded communications middleware.

Middleware is aimed at masking the differences between operating systems and processors and providing a single environment for all applications. It achieves its goal by encapsulating operating system API and services and providing its own API to the software developer.

The middleware-based approach suffers from the same problems encountered in virtual-machine-based designs, but to a much less degree. It inflicts less pressure on the processor than a virtual machine does.

To get through limitations forced by this approach, developers divide their code into two parts. One part is the middleware-based code incorporating a portion of the logic which is independent from the underlying OS and processor. The other part encompasses the code which is dependent on the processor and peripherals and communication interfaces or the code that needs to use a specific feature of the operating system on which it runs.

The first portion of the code can be re-used with no, or minor, modification in similar projects, while the second portion is solely developed and optimised for the current project.

In this sense, advantages of both virtual-environment-based approach and real-environment-based approach are present in this technique, while the disadvantages have been alleviated to a high extent.

From these statistics, currently the majority of companies that use middleware have developed their own. This is not unlike the early days of using embedded operating systems when many companies did, in fact, ‘roll their own’.

Over time, specialised third-party vendors emerged and the market moved away from in-house developed OS. It can therefore be expected that, like the OS market, a handful of options will end up being used for the majority of projects.

These possible solutions appear across a wide range of embedded applications. Radio communications systems suffer even more complexities. Wireless links, by physical nature, are more vulnerable to natural events or intentional disruption by intruders. These characteristics must be handled and this leads to complex protocols which can be hard to implement.

To make matters worse, wireless protocols are constantly modified to better reflect the improvements made to the field of wireless communications, which adds to the maintenance workload.

These days, however, the trend is towards more complete solutions. Development is often not viewed in isolation but as one part of the product life cycle. To cover the product life-cycle tools, middleware and life-cycle management all require consideration.

The successful use of software and system life-cycle management tools is a critical requirement for companies across every market segment. and systems manufacturers will require new solutions that help create, test and manage the designs necessary to power the next generation of embedded and enterprise systems.

The trends towards cohesive solutions, rather than isolated elements, combined with the increased portion of project spend on software vs hardware has been driving recent changes in the embedded suppliers market. Hardware players such as Intel and Cavium have acquired software vendors Wind River and MontaVista. It will be interesting to see what the future holds.