Optimizing ARM Cortex-A9 support in Windows Embedded Compact

A Discussion of random hangs and other issues using Windows Embedded Compact on Freescale i.MX6 Application Processor and how they were solved

By Adeneo Embedded Engineering Team
Rev 1.0, November 2014

Summary

Over the last year Adeneo Embedded has been confronted with reports of random processor deadlocks and operating system crashes from customers using our Freescale i.MX6 Windows Embedded Compact Board Support Package.

The random hangs and other issues surfaced while testing the devices comprehensively, i.e., regular CTK test passes did not bring out the failures to occur.

A dedicated team of senior engineers in Adeneo worked with a number of our key customers to analyze and solve the issues across the board. With the latest version of the Adeneo i.MX6 Windows Embedded Compact (7 and 2013) BSPs we are confident this has been achieved.

This white paper is about the investigation and shares some of our discoveries. All information in this document applies to Windows Embedded Compact 7 and 2013 as well as all variants of the i.MX6.

Format of the investigation

Based on the problem reports from the field it was complex to identify a single component in a system as the culprit, so we decided to use a formal and broad process to investigate the situation.

- A formal code review was done for the BSP code, Microsoft kernel code and customer application code;
- Lauterbach JTAG hardware debuggers were used to capture all available processor data at the time of crashes;
- Microsoft's kernel team assisted with all questions around the Windows CE kernel;
- Customer's engineering teams with their specific knowledge of their application developed test applications to replicate the problem more easily
- Freescale support engineers assisted with all questions around the silicon.
- Adeneo engineers redesigned the BSP from the ground and optimized it for i.MX6 and Cortex-A9 architecture

In summary, the collaboration of multiple companies, and more important, a diverse group of dedicated individuals with unique value add provided the comprehensive technical coverage to develop the solution to this complex problem

History of the i.MX6 BSP

Starting point for the i.MX6 Windows Embedded Compact BSP were earlier BSPs for other application processors from Freescale like i.MX5x series and back to i.MX2x series. On the OS side the history goes back to Windows CE 5.

The good thing with Freescale application processors is that they share peripheral IP blocks across a range of processors, so developers can share and reuse a lot of code. This was very helpful in the beginning to get a BSP working on the new i.MX6 SoC and get projects started. However, the i.MX6 with its multi-core Cortex-A9 architecture which made is challenging to reuse the code designed for single core Cortex A8 or ARM9 CPUs.

In particular, cache management, multi-core support and memory configuration were the areas where existing Cortex-A8 and ARM9 code first was able to get enablement possible, but then failed in long-term stability tests.

Cortex-A9 Architecture

The Freescale i.MX6 Application Processor is an implementation of the ARM Cortex-A9 and ARMv7 Instruction Set architecture. This powerful architecture provides a number of features to improve the processing performance, but requires special attention when developing system software.

The i.MX6 provides up to four cores in a symmetric multi-processing configuration under Windows Embedded Compact.

Some of the stability affecting features addressed during the investigation are:

- Speculative load and execution
- Speculative table walks
- Branch prediction
- Out-of-order execution and instruction reordering
- Parallel internal busses
- Multiple internal buffers and caches
- Multi-core coherency
- L1/L2 cache operations
- Abort handling

As part of our code review we identified shortcomings in existing code to correctly configure these features and take proper advantage of them. All code was verified with the ARM architecture documentation and updated to follow the latest recommendations by ARM. Freescale engineers helped to understand implementation details where ARM documentation is vague as it leaves some freedom to silicon vendors how to implement a feature.

All errata documents from Freescale, ARM and other IP vendors were reviewed, and we made sure all applicable fixes or workarounds got implemented in BSP or kernel code.

In discussion with customers we decided on a good working configuration for the i.MX6 processor that focuses on stability without compromising performance.

In multi-core configurations we updated the code to operate all available cores in the same configuration at all times. A critical area was power management code to reapply the same settings when coming back from low power states.

Memory Configuration

Cortex-A9 provides a powerful memory management unit that allows it to implement a virtual memory system that operates the device in multiple modes, isolates application processes from each other and provides layers of protection and security.

Looking at the memory space, we have several types of memory with this architecture. We focused on:

- Normal memory
- Device memory

ARM architecture has a flat unified memory address space as compared to x86 architecture where we have a memory address space and an I/O address space. This means all our peripheral registers and other I/O addresses are mapped into the same address space together with RAM and ROM (memory-mapped I/O). By default, this is nothing new and not a bad design as it makes things easier for software and hardware developers. In previous versions of Windows CE and other OS all addresses where treated a normal memory and the only difference between RAM and I/O was to set the non-cache flag in the memory properties for I/O. For architectures up to Cortex-A8 this was enough to ensure a stable operation of a system.

In particular with Cortex-A9 speculative engines the legacy approach causes problems. While some speculative features of the cores can be disabled, speculative table walks (which implicitly do speculative loads) can’t be disabled – for Normal Memory. So with Cortex-A9 it is necessary to use the extended access permission features of the architecture and configure all I/O memory as device memory. Device memory amongst others has the no-execute flag set in its properties (XN flag), and the processor doesn’t touch it during speculative operations. Under heavy load and stress this becomes an issue as the processor does more speculative operations per time and the chance to touch I/Os grows. It was one of the main reasons for crashes and deadlocks.

For Windows CE, Microsoft introduced a new way for OEMs to report available memory to the kernel with Compact 7. However, since the issue described above is not an issue on x86 and older ARM architectures, the i.MX6 BSP inherited the old reporting style from its ancestors.

With the legacy memory reporting the OEM fills a memory mapping table with the information about available memory and provides that to the CE kernel during startup. The kernel then creates the initial MMU page table with a cached and a non-cached entry per memory block from the OEM. For the MMU everything is normal memory.

The new WEC7 model works with two tables, the old one for RAM and ROM, and a new one for I/Os (device table). All blocks in the device table are configured as device memory in the MMU and are protected then.

This sounds straight forward, but the devil is in the details. The new model changes the way BSP code can use address translation during the early boot phase. Functionality in the startup code and the KITL component had to be updated in order to work with the new model and allow parameter transfer from boot loader code to OAL code. It is also not well documented and required kernel code reviews and discussions with Microsoft kernel engineers to fine tune this part of the code and optimize it for i.MX6. Another issue was that internal SRAM of the i.MX6, which in the first place appears as part of the processor’s I/O space, and so ended up in the device table. However, the internal RAM is used in low power modes to run power-management code while external RAM is in self-refresh, so it has to be mapped as normal memory without the XN flag set. After all, it wasn’t a trivial piece of work.

Synchronization Barriers

Due to the above listed enhancements in Cortex-A9 it is necessary to set synchronization points in the flow of operation at which the processor and all memory has a known state and is in sync. This is especially important when updating processor configuration or during context switches in the OS.

Through code reviews of OAL and kernel code and in discussions with Microsoft we updated the BSP to meet all ARM requirements and fine tune the interfaces between kernel and OAL to provide optimal performance.

Errata

During the investigation we spent time on errata for the i.MX6 and its various IP blocks. BSP and kernel code where intensively reviewed for each erratum, if they are affected and a fix or workaround is necessary to be implemented.

As part of this we also looked at the all software implemented BSP for i.MX6 (by Freescale) and its change log to double-check that we didn't miss anything.

Several critical errata were identified as missing in the code and implemented during the investigation. Three of the necessary code changes were in Microsoft kernel code and required a kernel update. Adeneo Embedded implemented these modifications in the kernel, tested the updated kernel in our test lab as well as with selected customers in the field, and then submitted the kernel change requests to Microsoft to formally release the update through the Windows Embedded Update mechanism.

Cache Management

The i.MX6 implements the Cortex-A9 architecture with an internal L1 data and instruction cache and an external L2 unified cache. Internal L1 means the L1 RAM array is located inside the ARM MPCore IP block, and each Cortex-A9 core in the MP cluster has its own L1 cache. External L2 means the L2 RAM array is located outside the ARM MPCore IP block but inside the SoC and connected to the internal AXI bus. Both RAM arrays are not accessible through processor load/store instructions.

Cache memory allows the system to keep often used data in memory with faster access but this requires to synchronize cache memory and external SDRAM so that observers outside the processor-cache block can see data changes.

When configured as SMP cluster some of the necessary L1 maintenance is done by the hardware cache controller. Since we may have up to 4 cores each with its own L1 cache, and a multi-tasking operating system which may assign the same execution thread to different cores due to context switches, it is necessary that all cores have the same synchronized view of the memory. This is done in hardware through the coherency unit in the MPCore as long as single addresses are affected. If the L1 needs maintenance as a whole software has to handle it.

Software also has to handle all L2 cache maintenance operations.

Cache maintenance gets invoked by the kernel normally, but there are a few situations where device drivers or even application software has to request cache maintenance. In any case, the OAL gets these requests and executes them. All cache related code in the OAL was updated and redesigned to meet the ARM architecture requirements and collaborate with the kernel in an optimized way. Shortcomings

Cortex-A9 support in the cache maintenance code and maintenance requests from drivers were another major source of instability in the initial BSP.

Some of the complications in this area are:

- Optimize L1 code to take advantage of the available hardware support
- Maintenance requests that include L1 and L2 require a specific procedure to make sure all levels of memory are in sync
- Maintenance requests can come from multiple threads and CPUs in parallel – L2 code has to be reentrant and multi-core safe
- DMA controllers work with physical addresses and do not know about caches; when DMA operations are used drivers have to make sure to request the necessary cache maintenance. Some instability with USB, SD and video operations were related to bugs in this area.

Build and Testing

As we learned at the beginning of this investigation that the CTK BSP testing failed to identify these issues we also reviewed our testing approach and implemented improved procedures. A new component in our testing is the stability lab, where we provide a dedicated set of hardware together with IT infrastructure to automate tasks and log results. With approval from our key customers we transferred customer test applications and applications that helped reproducing the issues into more generic test applications and added them to our portfolio.

Another lesson learned is that knowledge of customer use cases is important. We restructured our testing to be closer to real world scenarios and integrate feedback from customers directly.

A number of times during the investigation concerns were raised that the build process or the tools may be the root cause for some issues. Test teams reported different results based on where and how an OS image was built. We analyzed the tool installation and update process and the process to install OS bug fixes from Microsoft, but conclusion was that these observations were red herrings. But we used the knowledge gained from this part of the investigation to improve the build lab in Adeneo Embedded. We enhanced our infrastructure and upgraded tools so that it is easier for us to switch between versions and QFE levels of Windows Embedded Compact and our BSPs.

Commitment

This investigation was done over a period of about 8 months, and Adeneo Embedded put a committed effort into it to solve the problems. A core team of engineers worked fulltime on it while an extended group of engineers was available to support where needed (test, build, debugging, applications,..). The problems we had to solve here were not trivial; at times it was like a wild roller coaster ride.

But as a result we have a Freescale i.MX6 Windows Embedded Compact board support package by Adeneo Embedded with significantly improved quality and optimized for this system-on-chip. This brings out the benefits to all customers running Windows Embedded Compact i.MX6 and future CPU architectures on similar ARM cores on WEC7 and WEC2013.