a

A secure, field upgradeable operating system architecture for Blackfin Microprocessors

David N. Kleidermacher
Green Hills Software, Inc.
30 W. Sola St.
Santa Barbara, CA 93101, USA
davek@ghs.com

Jack Greenbaum
Green Hills Software, Inc.
30 W. Sola St.
Santa Barbara, CA 93101
jackg@ghs.com

ABSTRACT

Analog Devices' Blackfin processors implement a very flexible instruction set architecture geared for both control and signal processing applications. With dual MACs, a Memory Management Unit (MMU), and a control friendly instruction set, this is truly a flexible microprocessor. In particular, the inclusion of an MMU in a low cost, high performance processor with automotive peripherals has lead to the adoption of Blackfin in many automotive in-cabin applications. Such applications require the ability to upgrade after delivery. This paper will present an architecture that enables secure partitioning and updating of fielded systems with augmented or entirely new features without compromising reliability or security of the existing platform.

FIELD UPGRADE REQUIREMENTS

The requirements for field upgrade include the ability to replace existing functionality, and to add functionality. The upgrade itself may come from a variety of sources: locked features shipped with the unit, read from removable media, or transferred via a wired or wireless network. It is absolutely unacceptable for the upgrade to compromise the system in any way. In the in-cabin automotive designs where Blackfin is popular, such a failure could even compromise driver safety. To bring the features that consumers demand at ever lower costs, automotive engineers are designing with high levels of integration. No longer are there separate hands free kits, navigation systems, and multimedia systems. These capabilities are merging into one box. Care must therefore be used when updating any function because a problem with the upgrade can potentially disable all functions. For example one simple type of upgrade is loading a new navigation system database when crossing into a new country or region. If the upgrade caused a failure in a dashboard display or crashed the audio entertainment system causing a loud noise in the cabin, then a driver could become distracted. It is imperative that field upgrade of one system not affect the operation of others.

The field upgrade architecture described in this paper is based on the reliability and security model of the underlying RTOS. The RTOS must provide partitioning in time and space of processor resources. That is, the upgrade system must ensure the resources that existing pieces of the system depend on -- including CPU time and memory and peripherals -- are not compromised. A new audio codec must not be allowed to consume resources such that the navigation system no longer has enough processor time allocated to it to provide timely screen updates. A wireless Internet browser must not consume so much memory that the Bluetooth hands free capability no longer functions. If the underlying RTOS does not provide those resource guarantees in time and space, then no reliable field upgrade solution can be provided.

OVERVIEW OF BLACKFIN

Blackfin is a series of 32-bit microprocessors which combine signal processing and control processing in the same instruction set. Unlike some other "hybrid" microprocessors, such as the Starcore from Starcore LLC, the Blackfin provides a memory management unit that enables a system robustness capability not typically seen on digital signal processors. It is also simpler to engineer a solution on the single core Blackfin than on multi-core parts with both GPP and DSP processors such as TI OMAP.

Energy Efficiency

Like many microprocessors that provide signal processing capability, the Blackfin targets power-

sensitive applications such as automotive body electronics and digital entertainment and telematics systems, consumer multimedia devices, and cell phones. However, the performance to power ratio is what sets Blackfin apart. Able to achieve 750 MHz operation at a mere 1.45 volts, benchmarks have shown the Blackfin to outperform competitors by a comfortable margin in energy efficiency benchmarks. According to independent digital signal processing consulting firm, BDTI, the Blackfin BF533 part is roughly three times more energy-efficient than the Texas Instruments' TMS320C5501 and nearly four times more efficient than the TMS320C6414T [1]. In these comparisons, BDTI selected the most energy-efficient processors from each family.

Cost Efficiency

Another claim to fame of the Blackfin is its low cost relative to competitors at similar performance and power grades. BDTI reports that the Blackfin BF531 is 1.5 times more cost effective, in terms of cost-performance ratio, than the TMS320C5501 and nearly three times more cost effective than the TMSS320C6410. Blackfin processor prices start as low as $5 for 400 MHz parts in 10,000 unit quantities.

Memory Management Architecture

Like other high-end control processors, the Blackfin supports user and supervisor operating modes. In user mode, system resources are protected from access, including all peripheral interfaces and memory management unit registers. This design is similar to other memory management unit-enabled processor architectures. The main difference between the Blackfin MMU relative to more expensive processors is that the memory management unit does not support address translation; it only supports memory protection. Address translation is used to implement virtual memory capability on traditional protected mode operating systems. Therefore infrastructure for field upgrade must take into account the physical mapping required of downloadable software.

APPLYING SECURE RTOS TECHNOLOGY TO BLACKFIN

The Green Hills' INTEGRITY real-time operating system is well known for its reliability and security, fast real-time performance, and royalty-free business model, all characteristics that make it a natural fit for next generation power-sensitive applications running on the Blackfin. INTEGRITY was modified to be able to run on the Blackfin despite the lack of address translation capability. To date, INTEGRITY is the only real-time operating system that is able to take advantage of the Blackfin memory management unit to provide protection between user-mode applications and for the operating system itself.

The memory management unit is initialized and enabled early in the kernel's boot sequence to maximize the protection of early startup code. An initial large memory page is allocated for kernel execution. All code and data within this region are protected from user applications. A side effect is that no protected applications may be located within this memory page. In fact, this memory page is locked within the processor's translation lookaside buffer, called the CPLB on Blackfin. Therefore, kernel code and data can not be evicted by the CPLB replacement algorithm, ensuring fast execution of the kernel and interrupt handlers and ensuring system real-time response and determinism.

With the Blackfin architecture, supervisor mode is subject to the access policies enforced by the memory management hardware. This has the advantage that the kernel can be safe-guarded, for instance, from writing to addresses configured as read-only memory. However, protection enabling in supervisor mode has the disadvantage that even kernel execution could be subject to accessing memory ranges not described in the current CPLB table, thus causing kernel execution to be interrupted by CPLB miss handlers. CPLB miss exceptions must be uninterruptible for at least a portion of their functionality in any implementation of a protected operating system running on the Blackfin architecture, which means that CPLB miss handler execution increases interrupt latency. Therefore it is desirable to reduce the number of CPLB misses as much as possible.

RELIABILITY ENABLEMENT

The INTEGRITY real-time operating system uses the memory management and exception handling architecture of the Blackfin to implement the following reliability and security policies:

- Secure service calls

- Secure interprocess communication
- Memory protection
- Memory partitioning
- Real-time response
- Execution time partitioning

Secure service calls

Blackfin provides a user-mode exception instruction that forces a transfer into the supervisor mode. This mechanism is used to implement kernel service calls. INTEGRITY uses object descriptors instead of physical addresses for references to all kernel-managed objects. The combination of these opaque descriptors and the trap call interface enables the INTEGRITY kernel to verify all parameters of all service calls to ensure that a service call can not violate system security policies nor access internal kernel state. In contrast, other real-time operating systems use direct calls into the kernel for services and reference physical pointers to objects. In this model, erroneous or malicious application code can make improper calls into the kernel and obtain unauthorized access to internal kernel state, often with disastrous results.

Secure Interprocess Communication

INTEGRITY provides a secure, bi-directional communication channel, called a Connection, that enables tasks in the same or different address spaces to communicate. Unlike traditional message queues and other heavyweight protocols such as sockets, the kernel moves data directly between sender and receiver, without any intermediate copying or protocol overhead. Connection objects are securely assigned to the address spaces that need them, preventing communication from being tampered with by unauthorized applications. All communication transfers can be independently prioritized and scheduled, in much the same way as a thread's code execution is scheduled, enabling system designers to have fine-grained control of all activities performed by applications. Most other operating systems treat message transfers, even large ones, as overhead that can not be controlled at the same level as thread code execution. Using a resource manager provided with INTEGRITY, Connections owned by a component can live across the component's field upgrade without loss of integrity to the in-progress messages or Connection objects themselves. In a multi-processor distributed system, Connections can be used to seamlessly communicate between address spaces residing on different processors. A heartbeat is automatically built into the inter-processor communication to aid in failure detection and failover (Figure 1).

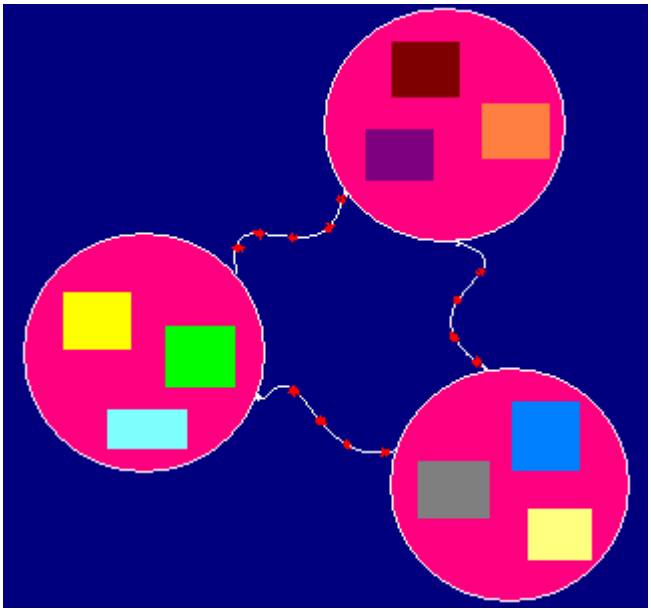

Figure 1 – *Interprocess communication with integrated fault detection*

Memory protection

The most fundamental requirement of protected mode operating systems is to protect the code and data of applications from unauthorized or unintended access from other applications. The Blackfin memory management unit is programmed by INTEGRITY to give applications access only to their own private code and data segments. Protected applications cannot circumvent the hardware enforcement to obtain access to code and data in other protected spaces, including the kernel. In addition, memory pages can be marked as read-only to provide read access to information that should not be writable, such as code and read-only data. Finally, the Blackfin supports separate CPLBs for instruction and data, enabling INTEGRITY to mark memory that is executable code distinct from other non-writable memory. Many other microprocessors are unable to distinguish between executable and read-only data, making them susceptible to stack overflow security attacks in which data segments are used to hold malicious code.

Memory Partitioning

INTEGRITY goes beyond memory protection to provide guaranteed memory resources to applications. A real-time operating system must use

memory to satisfy application requests to create objects such as threads, queues, and semaphores. Most operating systems employ what is called a central store to satisfy such requests. A poorly written or malicious application can perpetrate a denial service attack on applications, even those in separate protected address spaces, by overallocating and exhausting the shared store. INTEGRITY avoids these types of problems by providing quotas of memory to applications. In this system, the same offending application can exhaust its own quota but can not possibly affect the ability of other applications to obtain required object resources. Blackfin enables this partitioning architecture by ensuring that applications go through the secure kernel service call interface for allocation requests and by enforcing memory protection separation so that applications can not subvert the quota policies. Figure 2 shows an example Blackfin system divided into memory and resource-protected components. In contrast, Figure 3 shows a more traditional monolithic system in which applications share a single memory space, resulting in loss of robustness and difficulty in performing field upgrades.

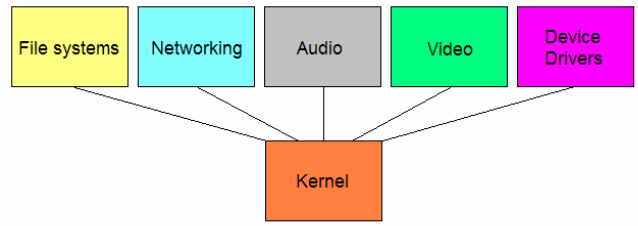

Figure 2 - *Partitioned System*

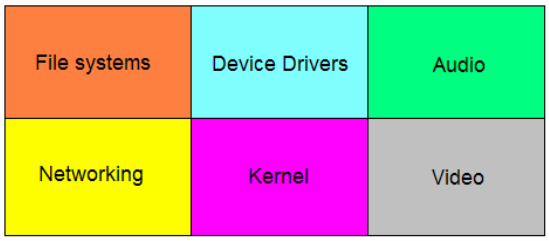

Figure 3 – *Non-partitioned System*

Real-time response

As signal processing and general purpose processing functions converge on the multifunction Blackfin microprocessor, designers face the challenge of ensuring that the real-time components of the system, such as audio, continue to meet their determinism and response time requirements despite the inclusion of heavy general processing loads such as wireless and wired networking.

INTEGRITY uses the Blackfin interrupt architecture to properly prioritize external interrupt requests and guarantee the minimum response time for the highest priority events. Unlike other real-time operating systems that completely disable interrupts during kernel critical sections, INTEGRITY achieves minimum interrupt latency by never disabling interrupts globally while processing service calls. In addition, INTEGRITY takes advantage of the Blackfin's individual interrupt enable bits to provide a prioritization between interrupts and threads. For example, the system designer can prioritize certain low priority hardware interrupts below that of software threads that are used to handle high priority interrupts. This type of interrupt architecture, lacking in other real-time operating systems, prevents a low priority interrupt from firing and delaying the processing of high priority threads that otherwise execute with all interrupts enabled. INTEGRITY's scheduler takes care of all the details of enabling and disabling selected priority levels to ensure absolute determinism, even under a heavy load of interrupts from multiple sources.

Execution time partitioning

INTEGRITY goes beyond the typical priority-preemptive thread scheduling found in real-time operating systems to provide an optional higher level scheduler for applications. Using this partition scheduler (Figure 4), system designers can assign fixed amounts of CPU execution time to address spaces that are already protected in the space domain by memory protection and partitioning. When a given partition is active, threads within the partition are scheduled according to standard priority-preemptive policies. However, threads in other protected applications must wait for their partitions to become active before they can execute. Like memory partitioning, INTEGRITY's time partitioning provides an added level of reliability for systems that mix multiple signal and general processing functions.

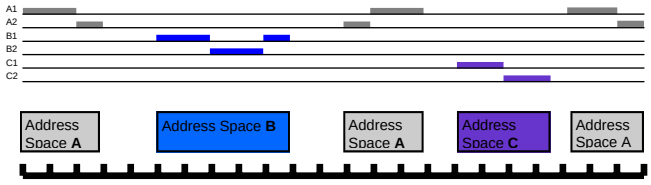

Figure 3 – *Partitioned Time Scheduling*

FIELD UPGRADE ON BLACKFIN WITH INTEGRITY

INTEGRITY provides field upgrade capability on Blackfin processors with a dynamic load mechanism. Protected applications can be dynamically loaded on a running system. These load modules can add completely new functionality to the system or can replace an existing module that has a flaw or limitation. In addition to loading of protected applications, INTEGRITY can be configured to allow the dynamic load of kernel code as well. Although this must be done carefully to ensure that the system cannot be subverted, kernel loads enable even the lowest layer of code in the system, such as interrupt service routines, to be updated in the field.

Blackfin's lack of address translation make protected application loading more difficult in that the load module must be linked to the specific physical addresses in which they are to reside. The system may set up a fixed pool of memory for dynamic loads and break this pool up into segments for each application. Load modules are linked for the appropriate segment and then loaded by the INTEGRITY dynamic loader. Although the system designer must manage these segments to avoid a failed load in which a module is linked at the wrong address, the ability to load protected applications to a running system provides reliable upgrades without jeopardizing the availability of actively executing portions of the system.

FUTURE WORK

To improve usability, the dynamic loader should be extended to enable the creation of address spaces from relocatable programs. The application build process would be similar to the current mechanism: the executable program consists of all the code and data from object files that make up the address space. However, the executable is not absolutely located at any particular address. Rather, the relocation information is retained in the executable file, and the target-resident loader is extended to include address resolution in much the same way as a host-based linker. Building relocatable programs in this manner is a standard feature of the Green Hills tool chain.

The advantage of this approach is that developers and system designers no longer need concern themselves with the location of general purpose applications (applications that do not need access to specific physical memory areas). As systems grow in complexity, managing a large set of applications, each with a different link map, can add significant overhead to the maintenance of the project. With a relocatable loader, the module load process entails simply finding some available memory large enough to hold the new address space, copying the code and data to the allocated area, and performing address resolution for all the code and data references. Since the velOSity microkernel already supports relocatable loading into the kernel AddressSpace, extending this mechanism to support the loading of relocatables into their own protected address spaces would be relatively straightforward.

CONCLUSION

Blackfin enables the convergence of signal and general processing applications onto a single compute platform that boasts impressive performance and power efficiencies. However, Blackfin goes a step further by incorporating a memory management unit that can be used to dramatically increase the overall robustness of these convergent systems. A memory protected RTOS which utilizes the MMU forms the foundation of a reliable field upgrade solution for Blackfin-based systems. Because current generation Blackfin cores do not support address translation within the memory management unit, a real-time operating system must use advanced and unconventional mechanisms in order to take full advantage of the reliability and security improvement potential and to ensure reliable field upgrades. The Green Hills' INTEGRITY real-time operating system has been specifically modified to bring its high reliability and security features to bear on the unique and powerful Blackfin architecture.

REFERENCES

1. *A BDTI Analysis of the Analog Devices ADSP-BF5xx-* http://www.bdti.com/articles/blackfin_summary_report.pdf, p. 3-4.