

# Application Programming Strategies for TI's OMAP Solutions

By Natalie Kloss

With great power comes great responsibility.”—Uncle Ben, a character from the movie, *Spiderman*.

This quote succinctly describes the situation of an engineer tasked with the job of designing the software for a system based on the TI OMAP processor. The difficulty presented by the unique dual processing capability of the OMAP is how to design software to use it effectively. This is particularly troublesome if the software in question is being ported from a system that is already working on traditional single-processor architecture. How can such software be adapted to effectively use the OMAP?

This paper explores these questions and gives some suggestions as to how either new applications can be created or existing code can be adapted to take advantage of the OMAP resources.

## OMAP from an application programming point of view

There is a great deal of documentation available from TI regarding the detailed system architecture of the OMAP processors. These details are of great interest to programmers involved in the creation of device

drivers or operating systems, as well as hardware engineers that create systems around OMAP processors. In general, from the point of view of an applications programmer most of this documentation can be summed up as “too much information.”

There are several points that are pertinent, though. The most obvious one is that there are two separate processors within this chip. There are communications paths between

them, but in general they are capable of (almost) completely independent operation. There are exceptions to this independence, which we will get into later.

These two processors have very different architectures. One of them is a relatively powerful general-purpose compute engine. This is the TI925 MegaCell side of the chip. This processor is based on the ARM9 design, as licensed by TI from

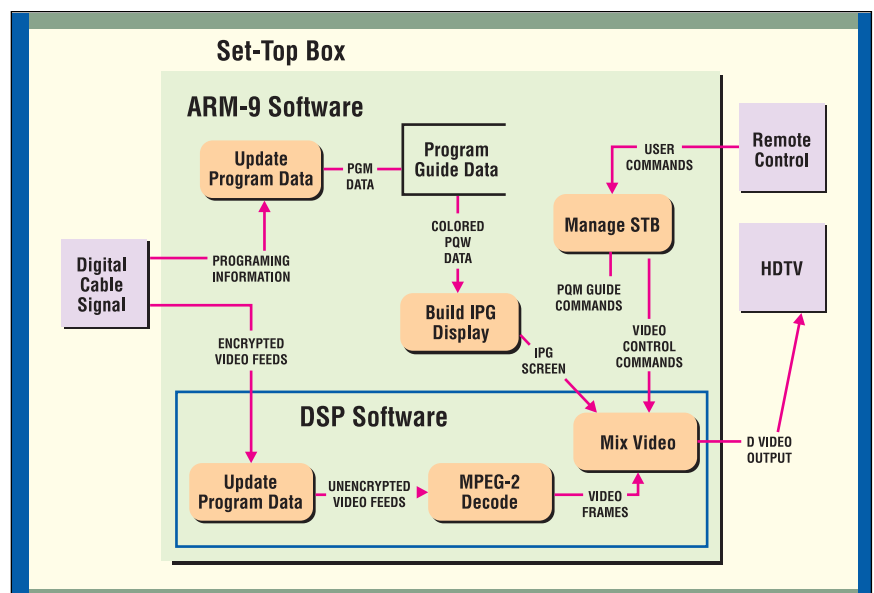


Figure 1: Example OMAP Partitioning for a Set-Top Box

ARM Ltd. It is worth noting that this processor is similar to but not identical with other ARM derivatives such as the StrongARM (designed by Digital Equipment Corp. and currently sold by Intel) or the ARM7 line, which was the predecessor to the ARM9. The most important point about this to the applications programmer is that the compiler must be aware of the particular processor to be used. This can be important, since the differences are in some cases subtle.

Programming the TI925 for applications in a HLL like C/C++ is literally just like programming for other UNIX or Windows-based systems, except that in most cases the TI925 systems will be running the code directly from code images embedded in the read-only memory rather than being loaded from disk. In fact, it will probably be relatively uncom-

Megastar3 CPU. This is the other half of the processing power of the OMAP line. This CPU is a Digital Signal Processor (DSP) that is highly optimized for performing repetitive complex mathematical operations. It is at its best doing things like convolution mapping or speech recognition. Anything that requires complex calculations on a large amount of data is grist for this mill.

This is not to say that the Megastar3 is incapable of performing as a general-purpose processor. This is the basic conundrum of the OMAP architecture. It is quite possible to do most of the calculations on the TI925 and much of the general-purpose processing on the Megastar3, but this would not be the most efficient use of the hardware.

In fact, efficient use of the hardware is the central question here. OMAP provides a tremendous

between application code and complex I/O processing, while the second might be more appropriate for a new development effort that can customize the code.

### Using the DSP as an I/O coprocessor

Many applications assume that the bulk of the control flow processing is based on a general-purpose processor. These applications often depend on data flows and intermediate data transforms being handled by smart, high-speed specialized hardware. These applications can map quite nicely into the OMAP architecture by replacing the specialized I/O processing hardware with firmware running on the DSP side of the OMAP.

This type of design cuts right to the heart of the question of hardware versus software as a design tool. For example, it is quite possible to embed modem protocols in a hardware design where they can be executed very reliably, but that design will not adapt to revisions in those protocols. A software-based approach on DSP hardware allows for revised code to be loaded onto the system after it is placed out in the field.

There are other advantages to this approach as well. Complex communications processing such as aggregation and correlation of multipath wireless data signals can be very difficult to program using fixed execution paths through hardware logic. A software-based approach allows the storage and utilization of relatively large arrays of data that can be used to best effect in these calculations. This can result in significant improvements in signal integrity and throughput in these systems while retaining transparency to the application code that utilizes this data.

It is a significant point that all of this data crunching happens independently of the TI925 processor. This leaves that CPU free to handle the user interface or perform con-

## Application programming on the OMAP will have a better chance of success if there is a consistent strategy involved.

mon for a disk to be present in the system at all. This means that more care must be taken to utilize resources efficiently—there will be no virtual memory to use if the physical memory runs out.

Other than the need to recognize the memory limits of the system the application programming environment will probably be very familiar. Systems that use Windows CE can be programmed in the familiar Visual Studio environment, while most other operating systems will utilize a derivative of the GNU C/C++ compiler. In either case, the programming tools will be very familiar even if the code being generated by the compiler is very different.

Speaking of different, there is the

amount of programming power, and it is up to the applications programmer to harness that power effectively. In the next section we will explore several ways that an application can be designed or modified to gain the benefits of this unique architecture.

### Some OMAP programming strategies

As with most complex tasks, application programming on the OMAP will have a better chance of success if there is a consistent strategy involved. The selection of a strategy for a particular project depends on the particulars of that project. Specifically, the first strategy might be appropriate for the port of existing code that has a clear separation

trol functions that may demand quick responsiveness. The memory architecture of the OMAP was designed so that there would not even be contention over shared memory resources in the system while both processors are free to do their jobs. When the DSP side completes an I/O operation it simply signals that fact through one of a series of mailbox interrupts to the TI925.

It is also worth noting that simple data transfers are not generally enough to keep the DSP side occupied. In fact, there is separate DMA controller hardware in the OMAP processors that handles the mass movement of data. Generally it is advantageous to move complex, repetitive data calculations into the DSP handling of the data stream.

One considerable advantage of this strategy is that it usually maps nicely for existing applications code. In fact, it may be possible to move extensive data handling applications into much smaller and cheaper hardware using this approach. It is quite possible that the DSP side of the OMAP could replace boards full of custom hardware logic.

### Splitting the application code

The previous approach works well in a situation where there is a requirement to move large amounts of data into or out of the OMAP processor. There are other applications, though, that need to perform complex operations on a standing set of data. These operations can benefit from the OMAP as well with a slight change of approach. In this case the idea is to move certain algorithms into the DSP to take advantage of the fact that it can perform them faster and more accurately.

This strategy holds the possibility of significant improvements in processor-intensive data processing. For example, many graphics algorithms entail tremendous amounts

of processing based on the characteristics of pixels in relation to their nearest neighbors. Operations like edge sharpening and other image processing functions can benefit greatly from the architecture of the DSP processor.

The trick in this type of strategy is to minimize the overhead involved in switching off between the two processors. The best bet is to reserve the DSP processor for extensive operations, especially ones that can happen asynchronously with respect to other operations on the TI925 processor. This is particularly true if extensive data sets have to be used for the calculations and the data is not directly accessible to the DSP. Pay close attention to the memory mapping capabilities between the two processors to avoid situations like this. Otherwise, the overhead involved in splitting the processing in this manner will remove some or all of the gains.

### Independent coprocessing

The previous strategies assume that the computing to be performed by the two processors is closely related. This may not be the case in all situations. In fact, it may be possible for the TI925 and the DSP to operate completely independently. If this is the case, then it is very straightforward to use the OMAP architecture. This is a fairly rare situation, however.

Much more likely will be the situation where there are two or more sets of operations that are lightly coupled. In this case there might be threads of operation on each processor that only occasionally have to synchronize operation. This can be done easily through the use of the mailbox interrupts. Note that there are mailboxes that are available to each processor that signal the other, so either side can initiate a message.

This type of design can also be

extended to multiple threads of execution running on each processor. The operating system support for each allows multitasking at the application level, so it is very easy to set up an independent thread. Note that multiple threads running on a processor will be sharing the resources of that processor, though. You will not get the advantage of running these processors on completely independent hardware.

Figure 1 shows an example of a system that has been partitioned for implementation with an OMAP processor. This example is a digital Set-Top Box (STB) for HDTV signals. This example is particularly apropos, since many early STB designs had trouble meeting real-time goals for MPEG decode, which is certainly a prime function for the device.

This partitioning will relieve problems such as those that plagued those early designs. Here the DSP is dedicated to the tasks of decryption and decoding, while the general-purpose processor is free to manage the device and respond to user devices. Note that the GP processor is also responsible for building the Interactive Program Guide screen, since this procedure is done relatively infrequently.

Granted, a complete STB design will be much more complex, but this example is complete enough to show the real synergy between the dual processors in the OMAP design. If the management portion of the design is relatively simple, as was true in many early STB designs, it may be possible to integrate this functionality into a single processor. This simplicity, however, is becoming much more the exception rather than the rule. ♦

For questions concerning Stellcom, please contact: *Natalie Kloss*, Director of Business Development, 512-370-3116, [nkloss@stellcom.com](mailto:nkloss@stellcom.com)

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

<b>Products</b>		<b>Applications</b>	
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>	Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>	Automotive	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>	Broadband	<a href="http://www.ti.com/broadband">www.ti.com/broadband</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>	Digital Control	<a href="http://www.ti.com/digitalcontrol">www.ti.com/digitalcontrol</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>	Military	<a href="http://www.ti.com/military">www.ti.com/military</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>	Optical Networking	<a href="http://www.ti.com/opticalnetwork">www.ti.com/opticalnetwork</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>	Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
		Telephony	<a href="http://www.ti.com/telephony">www.ti.com/telephony</a>
		Video & Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>
		Wireless	<a href="http://www.ti.com/wireless">www.ti.com/wireless</a>

Mailing Address: Texas Instruments  
Post Office Box 655303 Dallas, Texas 75265

Copyright © 2003, Texas Instruments Incorporated