INTERNET TELEPHONY

Configurable Open Source DSP Software for VoIP

Is it Right for You?

By Scott D. Kurtz

E-mail to a Friend Order Reprints Print TrackBack Bookmark Free Magazines eNewsletters RSS/XML

The time has come for Open Source to do for Digital Signal Processing (DSP) applications what it has already done for other software applications: allow developers to leverage proven software yet still customize it to their particular applications. DSP applications bring with them some unique challenges with respect to Open Source, but with unique challenges come creative solutions.

As product developers, what do we want to do? We want to develop a product that meets a set of requirements as efficiently as possible. Open Source Software is an excellent tool to have in ones toolbox. By making use of (presumably) proven software, enginee can reduce a products development time.

But what do we want to do? Imagine a time when it may be possible to generate efficient production-ready software by connecting together a bunch of blocks in a PC-based diagram editor. Yes, applications like this exist today for the purpose of modeling and simulation, but they tend to generate inefficient code and are not well-suited to generating code that is easily used in an embedded system.

Part of the difficulty is that we are dealing with two opposing forces generality and efficiency. As I type this article using a modern word processor, a good analogy stares me right in the face. I probably use fewer than 10 percent of the features offered by this word processor. The word processor software takes up quite a bit of disk space, and I can only guess how much CPU resources are used b even the simplest Windows-based application. Fortunately, disk space is plentiful these days and I have the seemingly limitless power a 3 GHz CPU at my fingertips. Unfortunately, resources are not so abundant in embedded and DSP applications.

While we may not be at the point where we can create efficient software using block diagram editors, at least one option does exist today to take us part way there in particular for VoIP DSP software. Before presenting this option, it is helpful to review traditional DSP solutions.

Traditional DSP Solutions
What DSP options have designers had available to them up until now? At one end of the spectrum is the fixed-function chip. Fixed-function chips are designed and optimized for a specialized application. While they are very efficient, they are also very inflexible They are available off-the-shelf, yielding a faster time to market for those who use the chips.

At the other end of the spectrum lies the programmable DSP. The programmable DSP can yield very efficient solutions also. After all, many fixed-function chips are actually DSPs that are bundled with application specific software. The difference is that when the design starts out with a programmable DSP, he or she is faced with the task of developing a considerable amount of software as well as the underlying DSP algorithms themselves.

Fortunately, there are other options somewhere between the two ends of the spectrum. Many designers opt to license the DSP algorithms from third-party companies that specialize in such things. The designers then are left with the job of writing the application code that makes use of the algorithms. Writing the application software itself is no small task, and the intense real-time requirements imposed on a DSP application certainly contribute complexity and difficulty to the problem.

Reference Applications and Reference Frameworks
In order to save designers time, a few DSP software vendors offer reference applications. The reference applications are offered primarily in C source code format. Since it is not reasonable to expect a reference application to be a better fit for a design than a fixed function chip will be, having the source code enables the designer to customize it and make it a perfect fit.

It is important to note the difference between a reference application and a reference framework. A reference framework tends to inclu a generic sample application, that takes data in, processes it with a sample algorithm, and outputs data. A reference framework is not designed with any application in mind. A reference application may start with a reference framework, but it is designed with a specific application, such as VoIP (define - news -alerts), in mind. For example, a reference VoIP application should incorporate suitable interfaces such as IP, PCM, and host interfaces. It should also include the algorithms typically used in VoIP systems such as speech compression, echo cancellation, conferencing, tone detection and generation, and fax relay. It should also support the channel configurations that are typical in VoIP gateways, PBXs, and endpoints. A VoIP reference application is therefore a far better starting point for a VoIP designer than a reference framework.

VoIP applications vary widely. Features that vary between VoIP applications include the number of channels per unit or chip, channel configurations, peripherals used for input and output, and required algorithms. A reference application that tries to be everything to everybody will include many features that a particular application may not need. Just like todays word processors, an application this general will take up more processor resources (CPU cycles and memory) than an application that that is more tailored to a specific configuration. This leaves the designer with the task of removing unneeded functionality from the reference application. On the other hand, if a reference application is more specific it may not include all the functionality needed by a particular application, leaving the designer to add functionality, which tends to be even more difficult than removing functionality. While the reference application provide a far better starting point than the aforementioned options, it still leaves something to be desired.

Configurable Reference Applications
This leads us to the Configurable Reference Application. A Configurable Reference Application is one that can be customized easily using a windows-based Graphical User Interface (GUI). The designer uses a configuration utility to select the features (algorithms, channel types, number of channels, peripherals) that he or she wants included in the target application. A fully featured configuration utility will even allow the designer to select the target DSP chip. Once all selections are made, the configuration utility modifies the application source code and project files, eliminating the unnecessary features automatically. The configuration utility then invokes the appropriate compile tools and generates a binary DSP-downloadable image.

If this were the end of it, we might still be stuck with some inflexibility. Even a full featured reference application cannot anticipate everybodys needs. Some applications may require custom algorithms or custom interfaces. A binary software image is what it is. It is r made to be changed. This is where Open Source brings its value back to the table. By having the reference applications source code, already customized by the configuration utility, the designer can make the last-mile changes that achieve the exact required functionali nothing more, nothing less.

A good reference application should go beyond the boundary of the DSP itself; it should also include software that simplifies the interface between the DSP and the host processor. This simplification is done by providing a software abstraction layer (in source code of course). The software abstraction layer provides a set of APIs that perform functions such as downloading the DSP software image, configuring the chip, setting up and tearing down channels, reading and writing packets, etc.

The designer builds the supplied API software into the host microcontrollers software. The microcontrollers application software calls tl API functions, which in turn call hardware specific drivers that carry messages between the microcontroller and DSP over the interface of choice. For example, some systems may communicate between host and DSP via a parallel host port interface while others may communicate via Ethernet.

Is Open Source DSP Right For You?
There are many tradeoffs to consider in choosing the right DSP solution: flexibility, efficiency, cost, time-to-market, and having control your product. Efficiency can be measured in a number of ways. In many applications, the goal is to minimize cost per channel. In other applications, it is more important to minimize total power dissipation or board space.

Cost includes two general components, recurring and non-recurring. Recurring cost includes the price of the chip (and any required peripheral chips), the software licensing fee, the cost associated with a smaller or larger printed circuit board, etc. Non-recurring cost includes hardware and software development cost and one-time software licensing fees. Yet another cost is opportunity cost. This relates to time-to-market. If a product takes a long time to bring to market, one must consider the revenue that is lost by not bringing th product to market sooner.

Having control of your product is not to be overlooked. Chips are discontinued from time to time. It is important to consider the likelihoo that a chip will be discontinued during a products lifetime, and the cost involved in replacing it. Another aspect to having control of your product is related to making changes. Even if a fixed function chip meets a products needs today, it does not allow for feature enhancements down the road.

The fixed-function chips tend to be the least costly for very low-volume products. Cost includes both recurring and non-recurring cost. Since the non-recurring cost is zero (assuming that you arent the one making the chip), that is reflected in a low cost. As production volume increases, however, it is often justified to spend some non-recurring dollars to lower the recurring cost. The fixed function chips also tend to be efficient because they are designed to implement a specific set of functions. It is important to qualify the statements about low cost and high efficiency of a fixed function chip. This is only the case if the chip has the exact functionality required by the user. If it has excess functionality, you end up paying for features and performance that you dont need.

Configured applications are nearly as efficient (arguably as efficient) and cost-effective as fixed function chips without the same qualification. They are far more flexible because they are configurable and because they are software-based and can be modified and updated. Since configurable applications run on general purpose DSPs, the likelihood that the chips may be discontinued is less than i the case of a fixed-function chip.

Programming the entire solution yourself gives you, the designer, ultimate flexibility, but the cost is enormous due to the significant software development effort as well as the opportunity cost.

The choice is, of course, yours. IT

Scott Kurtz is vice president at Adaptive Digital Technologies, Inc. For more information please visit the company online at www.adaptivedigital.com (news - alerts).

[RETURN TO THE TABLE OF CONTENTS]