



RoweBots
Research Inc.

White Paper – 3 July 2009

DSP Components and Embedded Linux

Author: Kim Rowe,
Founder, RoweBots Research Inc.

www.rowebots.com

DSP systems

DSP is more and more a part of systems rather than a stand alone element. These systems require lean product development or platform based approaches and industry standards for portability of applications. Through the use of these approaches, companies will maximize profits and minimize both risk and time to market.

Given a need for standards based approaches and open environments, how do semiconductor companies support lean development for systems which contain DSP and DSC components? How do they preserve customer's investments in applications? How do software and system providers augment these base solutions? What is the best solution?

Hardware and Software Architectures

Architecturally, we will assume one or more MPUs, MCUs, DSCs, DSPs or FPGAs at the core with some functions in hardware and some in software. **We will also assume that software is required to drive the main I/O components and some sort of control software is required.**

For these processors we will consider several variants for the software architecture. The cases are:

- single loop of control
- proprietary kernel
- open standards based kernel
- proprietary RTOS (kernel with I/O model, documentation, testing, ...)
- open standards based RTOS (GPL)
- **open standards based RTOS (GPL Free)**

Hardware and Software Architectures

continued

The first case of a single loop of control is very straightforward. If the system has a single external event to handle, this works well. A second event may be possible but past this maintenance is compromised,

The case of proprietary kernels and proprietary operating systems are very similar. Due to their proprietary nature, they don't offer an open standards based approach. This leads directly to lock-in to a specific vendor. Of course, many vendors favor this but it substantially limits future architectural choices, driving up costs and reducing profits.

Open Standard Based Software Architectures

Open standards based solutions involving a kernel or complete operating system can be considered together because one is a subset of the other in the marketplace. Generally, open standards based solutions have been limited to larger processors; however, today there is general support for MCUs,

The de facto standard for embedded systems today is Linux. Although many POSIX compliant systems and even windows based versions are essentially standards based, the clear trend today is to use Linux. It offers millions of lines of reusable code, embedded real-time options and a huge selection of I/O. *Figure 1* shows the software options positioning in the market.

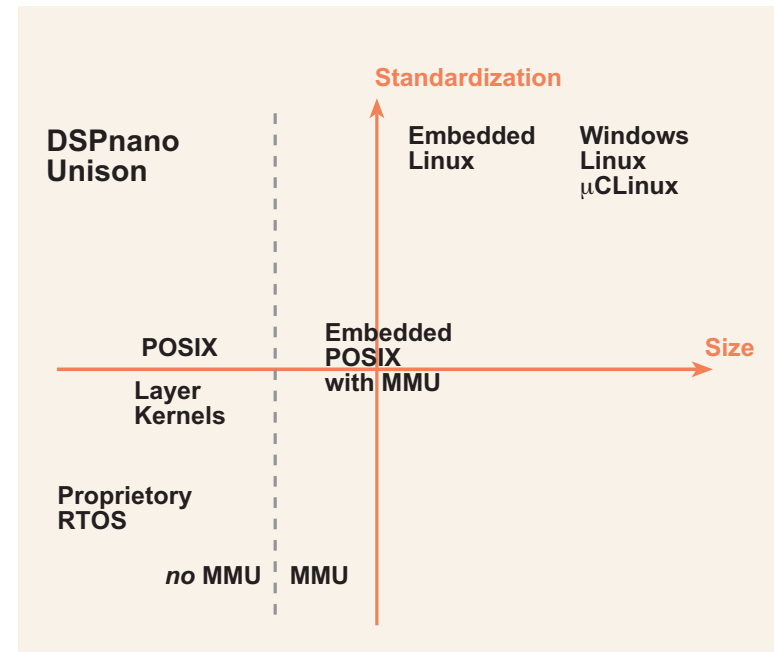


Figure 1: Operating System Positioning

MPUs and MCUs with MMU

On larger MMU capable processors, Linux is generally the operating system of choice. This choice is possible because the MMU essentially allows application developers to avoid GPL licensing. They don't have to share their applications with others and still benefit from open standards and open source community development.

Often, on larger MMU capable processors, there is a push towards virtualization. By running legacy operating systems in conjunction with Linux, application redevelopment can be avoided and costs reduced. This is an often used migration approach for very large systems.

DSCs, MCUs and DSPs

Typically DSCs, MCUs and DSPs don't use an MMU. For this reason, GPL causes problems for developers. They certainly don't want to give their applications to their competitors. The large size of Linux, its lack of modularity and GPL have made Linux and Linux variants a non starter on these processors until very recently. An Ultra Tiny Linux and POSIX compatible pair of operating systems from RoweBots now offer modularity, a GPL free license, open source and 100% Linux and POSIX compatibility for these smaller processors.

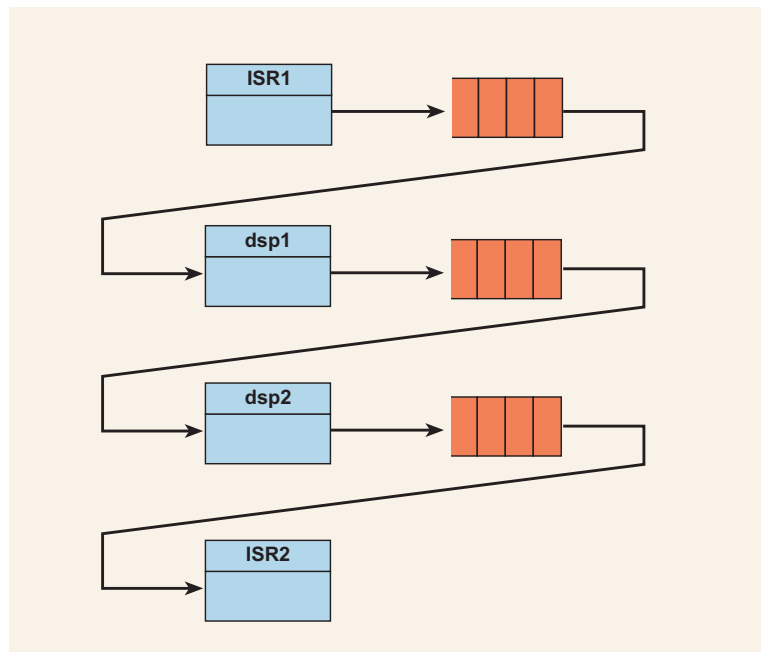


Figure 2: DSP Pipeline with two threads and two ISRs

DSP Pipelines Best Practices

The core approach to building a **DSP pipeline** (as shown in figure 2) in software rests on the following approach.

- Build a zero copy DSP pipeline.
- Allow threads to implement elements of the pipeline as components.
- Use message queues as inputs and outputs and pass only pointers.
- Use fixed size buffer managers which can be used by any thread.
- Implement standard algorithms which receive an initialization message, get blocks of data from the input queue, process in blocks and post the results to the output queue, managing buffers as required
- Allow all threads to run at a priority which works for the system, typically with the I/O related threads as the highest priority.
- Output to and input from acceleration hardware such as FPGAs and ASICs to optimize the use of hardware in signal transformation.

From this, it is easy to see that the core building blocks for optimal standards based DSP component implementation are:

- threads,
- initialization messages,
- queues for input messages,
- queues for output messages,
- ISRs
- and fixed size buffer management, on the software side.

On the hardware side the components are:

- FPGAs
- ASICs
- Special DSP/DSC/MCU accelerators.

DSP Pipelines Best Practices

continued

Figure 3 shows a standard software component with its inputs, outputs and buffer management.

Looking at the details of the POSIX and Linux compatible features required to provide optimal system level implementation; first, message queues with a queue of pointers to buffers is ideal. Ownership of the pointer to the buffer gives ownership of the buffer and it can be easily passed if required.

Next, each thread should have an initial message which configures the processing. For example, the initial message will specify the input and output queues, signal processing parameters like coefficients, algorithm details or options, and necessary buffer pools.

Third, fixed size buffer managers with getting a buffer and freeing a buffer callable from interrupt service routines (ISRs) are key. This is required to allow the components to manage buffers without overhead. Availability to manage buffers in ISRs is required to eliminate any buffer copying.

Priority based thread scheduling is used to make sure the critical components in the pipeline are processed in the correct order to maximize throughput. Often buffer shortages occur and system results should not be negatively affected by temporary data surges.

Fifth, ISRs are required to handle I/O at the end of the pipeline or midway in the pipeline if external hardware is to be used to accelerate processing. In the case of external

hardware, two ISRs would be used; one to send the data out to the external accelerator and the second to input the processed data.

Message queue post from ISR and message queue try_receive from ISR are also required to ensure that the ISR can get a buffer, fill it and then pass the data on for processing without copying. Similarly an ISR can get an output request from the queue without blocking.

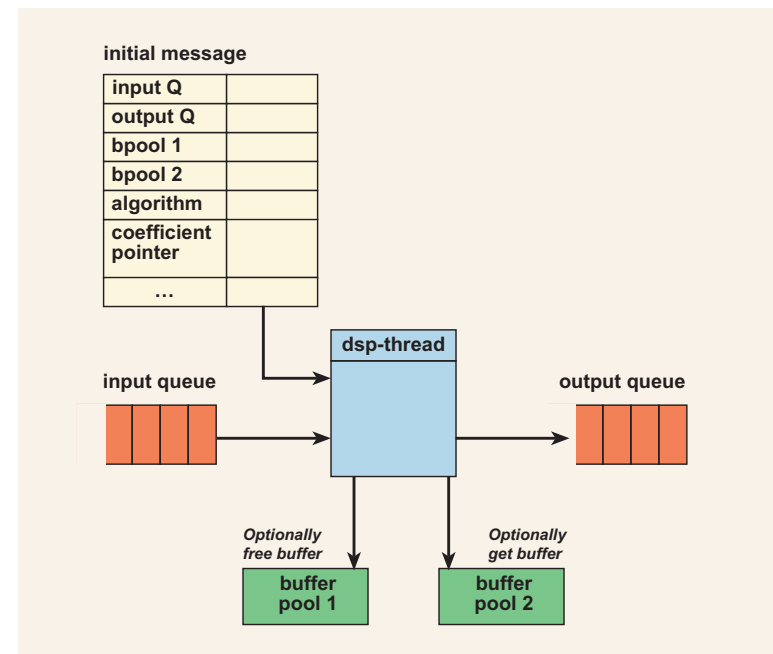


Figure 3: The simple structure of a DSP pipeline component

DSP Pipelines, POSIX and Linux

From the best practices for both DSP Hardware and Software Architectures, and DSP Pipelines we can see that implementation of DSP pipelines on POSIX or Linux systems is not only possible but a one to one mapping. The only calls which are not 100% POSIX and Linux compatible are the fixed size buffer allocators. These allocators can be implemented using malloc or free in ways to make them 100% compatible as well.

In this discussion of compatibility of DSP pipelines with POSIX and Linux, there is no restriction regarding the MMU. This means that applications can be developed on Linux or POSIX based systems and then ported from one to another – there is no lock in or restrictions with this approach.

Commercial Realizations of DSP Pipeline Capable Operating Systems

As discussed above, there is the possibility of using a broad set of Linux and POSIX compatible operating systems on processors with MMUs. As long as the operating system has basic POSIX calls as outlined above, the application can be programmed and quickly and easily ported to other systems.

For DSCs, DSPs and MCUs, there is two operating systems today that offers the modularity, tiny memory footprint, performance and 100% Linux and POSIX compatibility. These two operating systems are **DSPnano** (8/16 bit) and **Unison** (32 bit) from **RoweBots**. Both offer free

www.rowebots.com

development, open source, free non commercial distribution, patent free certification and a GPL free license.

The larger vendors involved in DSP and FPGA implementations offer both a proprietary offering and typically a Linux offering. Proprietary offerings try to lock customers in to a particular set of chips and the Linux options can be moved to many different underlying hardware architectures. With smaller chips, typically there is no operating system.

Summary

The choices for implementing standards based DSP pipelines as components are limited to Linux and POSIX compatible operating systems on MPUs or MCUs which have MMUs.

On DSCs, DSPs, MCUs and FPGAs without MMUs; there are two operating systems from **RoweBots** which offer the modularity, ultra tiny footprint, performance and 100% POSIX and Linux compatibility to be practical and problem free on these systems.

White Paper – 3 July 2009

DSP Components and Embedded Linux

Author: Kim Rowe,
Founder, RoweBots Research Inc.

Contact Information:

Kim Rowe, Founder

sales@rowebots.com

+1 519 208 0189

+1 519 498 6917



RoweBots
Research Inc.