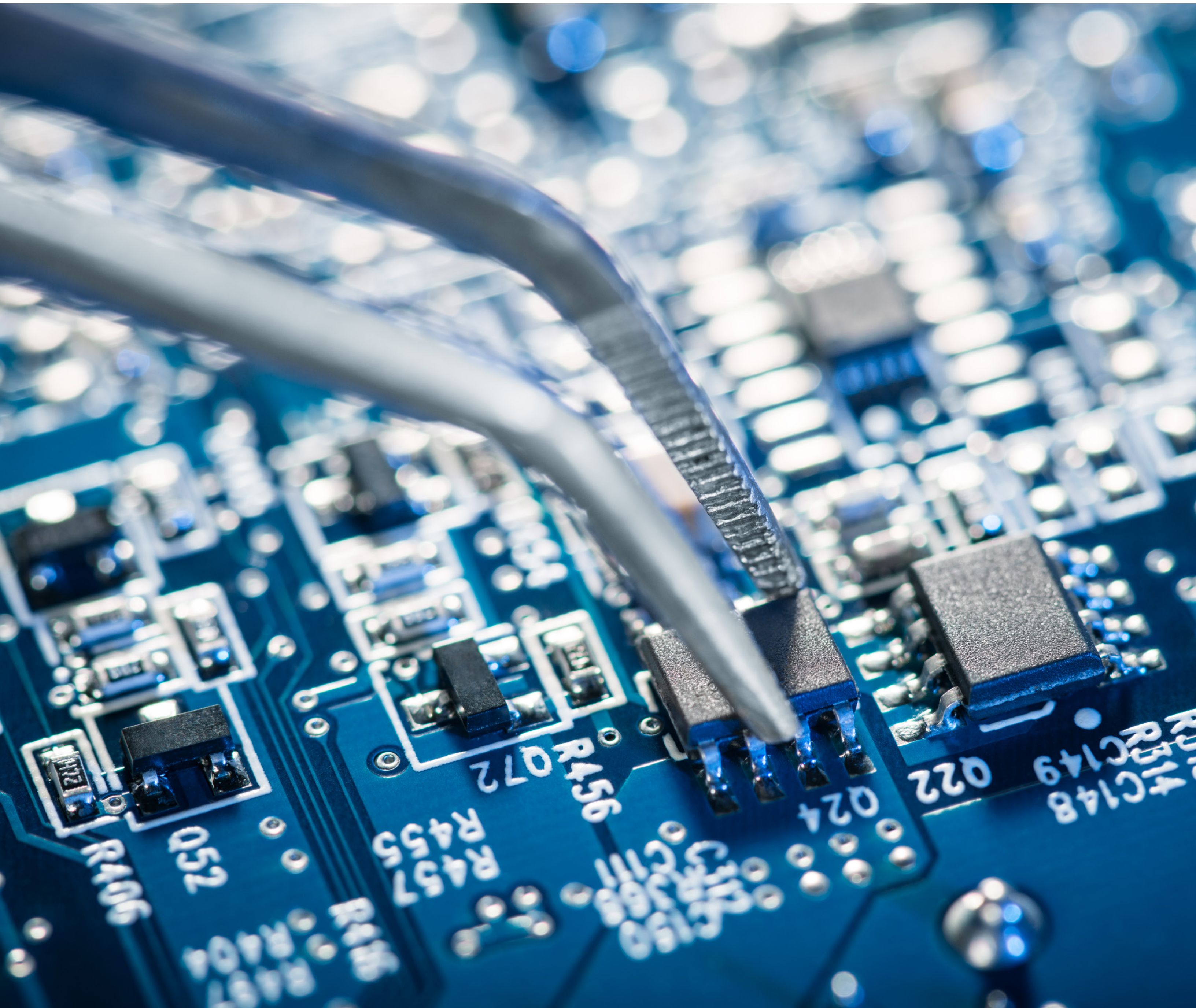
How to reduce the bill of material costs with digital signal processing

By Jacob Beningo,
Embedded Software Expert

White Paper



Contents

The need to decrease the bill of material (BOM) costs in embedded products is being driven by the need for high volume, low-cost sensor systems. As IoT devices become more sophisticated, they require developers to utilize digital signal processing to handle more features within the product, such as device provisioning. In this paper, we will examine how digital signal processing (DSP) can be used to reduce a product's cost.

1. Introduction: technology trends are moving processing to the edge of the network

The embedded systems industry is currently undergoing a significant transition that is being driven by the need to have intelligent, real-time, edge devices in the IoT. As systems need to do more at the edge to enable more compute performance, it's forcing devices to do far more in software than ever before. In the past, data could be streamed to the cloud for processing and analysis, but as intelligent real-time systems begin to fill the landscape, that processing needs to be done at the edge.

"In the past, data could be streamed to the cloud for processing and analysis, but as intelligent real-time systems begin to fill the landscape, that processing needs to be done at the edge."

It's not just intelligence that is driving more to be done in software at the edge. For example, a significant pain point facing IoT device manufacturers is how to provision their devices. Today, this is a manual and labor-intensive process. The end user must first set their device as a Wi-Fi hotspot. Next, they use their phone or computer to connect to the device. Finally, they must enter all their credentials into the device and restart it. A process that may seem simple but is challenging to many end users.

An interesting alternative to this process is to use **Chirp**. Chirp uses audio tones to transmit credential information to an edge device to provision it. This eliminates all the extra steps necessary to configure an IoT edge node. Devices that will be using this technology will need to be already capable of supporting digital signal processing in order to process the audio tones.

Companies are also deploying a wide variety of small sensing devices in large volumes. To be successful, these devices don't just need to decrease their overall BOM costs but also need to be flexible to handle a wide array of analog sensor data with minimal changes to the hardware. To meet these needs, developers can leverage DSP to convert analog circuits into software algorithms that have similar characteristics to the original circuit.

In the past, converting analog circuits into software comprised of many challenges, including:

- Expensive dedicated DSP processors
- Time-consuming software development
- Longer testing cycles
- Required expert knowledge

"… developers can leverage many different solutions to help them quickly, easily and cost-effectively utilize DSP in their designs."

Today, developers can leverage many different solutions to help them quickly, easily and cost-effectively utilize DSP in their designs. These include:

- Using a microcontroller with DSP extensions
- Utilizing modeling software that can generate the software algorithm
- Accessing free DSP libraries such as CMSIS-DSP

In this paper, we will discuss how developers can convert analog circuits into software and the many advantages that go with it.

2. Evaluating the benefits of digital signal processing

Before discussing how to convert an analog circuit into software, it's important to realize the benefits that DSP can have other than reducing BOM costs. These benefits include:

- Decreased product dimensions
- Product flexibility
- Shorter design cycle
- In-field adaptability
- Consistent product characteristics

There are two benefits in this list that warrant further elaboration.

First, when a product uses analog circuitry such as an active filter, the circuits need to be designed to take in to account aging and environmental factors. For example, if the product is placed in an environment where it's 85C, the analog components will have a different value than if you place them in a -40C environment. The filter may behave entirely differently at one temperature than at another. Another example is that most analog component values start to drift as they age. The filter output will change as the components age.

DSP removes this inconsistency and ensures the same device behaves the same under different environmental factors and over time. There is nothing that can physically change. Using software also ensures that the same behavior will be seen across the entire line of manufactured devices. In IoT devices, products may be deployed in the field for a decade and sometimes longer in many cases. It's critical these devices operate the same after ten years as they did on the first day.

The second benefit that is also important for us to discuss is product flexibility. A product that uses analog circuitry has a fixed operational behavior. If the product is deployed in the field and then it's discovered that the filter needs to be changed, the device needs

"Using DSP can dramatically increase flexibility and reduce maintenance costs by storing important filter parameters in EEPROM or flash."

to be recalled modifying the board physically. Alternatively, if the product needs to be adapted for a new sensor, industry or feature set, the physical hardware needs to be updated.

Using DSP can dramatically increase flexibility and reduce maintenance costs by storing important filter parameters in EEPROM or flash. While in the field, if adjustments need to be made to the filter, a few changes to the configuration parameters is all that's necessary. If the product needs to be used in a new application or industry with very different requirements, there's no need to change the hardware. Just update the software filter that's needed.

DSP software can be an extremely powerful tool for developers who understand how to utilize it properly. To learn more about the benefits that DSP offers, check out my latest blog: 5 benefits to replacing analog components with DSP software.

3. Converting analog circuits to software

One major challenge that may make an engineer leery of switching from an analog circuit to software is the knowledge gap necessary to make the transition. How can a developer make sure that the performance characteristics from the analog circuit are successfully met in the software algorithm? On top of that, many embedded developers aren't experts in DSP and designing an algorithm could be difficult.

It turns out that the challenge is not as difficult as it first seems. There are several ways that developers can convert their circuits to software. The most efficient method that requires the least level of expertise is to use modeling software such as MathWorks Matlab or Advanced Solutions Nederlands (ASN) Filter Designer. These tools allow developers to design the filter that they need and then automatically generate the software that can be deployed to an embedded system. Let's look at an example.

The circuit in Figure 1 is an active pre-emphasis filter that is commonly used in speech applications and that with very few modifications is used in a wide range of analog sensor filtering hardware. It's probably one of the simplest filters that anyone would want to convert into the software. It consists of:

- A single operational amplifier
- Three resistors
- One capacitor

On its own, this circuit represents approximately $0.20 when purchased in volumes of 1,000. It's a low-cost circuit, but $0.20 taken in large volumes can quickly add up significantly especially if the circuit is duplicated multiple times, which is common.

The BOM isn't the only cost that should be considered here but also the cost for the space on the printed circuit board (PCB), the extra energy consumption from leakage currents that could potentially allow for a smaller battery, and even the fact that a smaller PCB can result in lower weights for shipping the product. These are just a few considerations that affect cost.

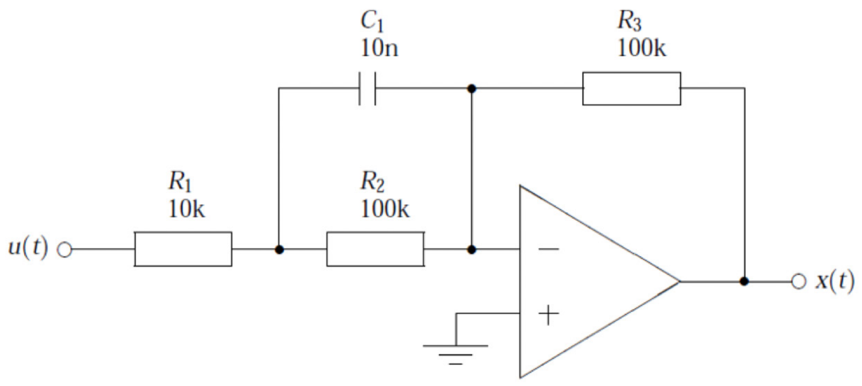


Figure 1: A simple active pre-emphasis filter that is common in many sensor application circuits. (Image Courtesy ASN)

There are several steps that a developer should follow to convert their existing analog circuit into the software. The steps include:

1 - Mathematically model your analog circuit
2 - Determine the digital sample rate
3 - Create a digital model that reasonably matches the analog circuit
4 - Verify the model
5 - Automatically generate CMSIS-DSP compliant software
6 - Integrate the generated DSP software into the product code base
7 - Test and verify the software solution provides the same results as the analog circuit

As an example, the pre-emphasis filter could be modeled using ASN Filter Designer. Once done, the difference between the filter gain and phase is nearly negligible as seen in Figure 2.

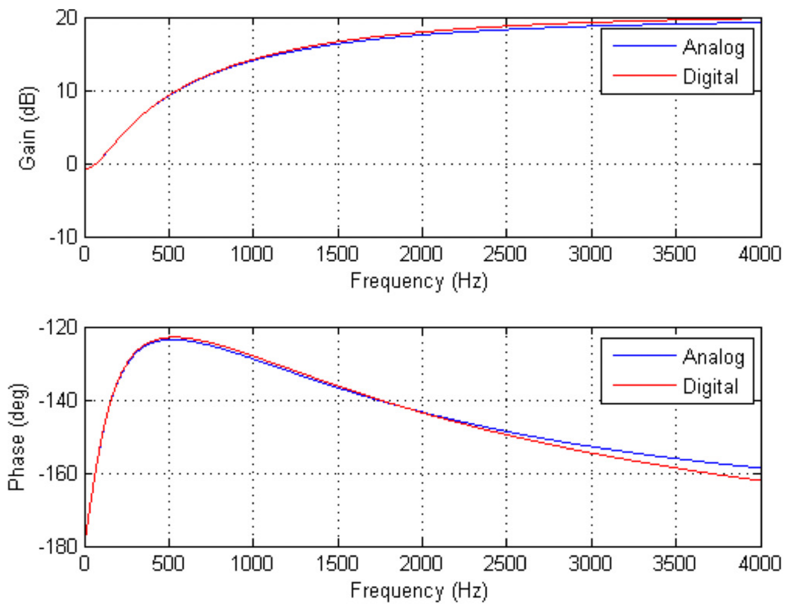


Figure 2: Modeling the gain and phase behavior of the pre-emphasis filter compared to the DSP software implementation. The two implementations are nearly identical. (Image Courtesy ASN).

If you are interested in learning more about how to convert an analog circuit into software, please read the blog: Deploying legacy analog filters to Arm Cortex-M processors.

4. Selecting the right DSP processor

As we have seen, there can be quite a few benefits to converting analog circuits to software and depending on the end application, the BOM and manufacturing savings can be significant. Unfortunately, selecting the wrong DSP processor solution can easily wash away the cost benefits. (There are still plenty of fringe benefits to make the switch well worth it).

There are primarily two choices that developers have for DSP architectures. First, there is a dedicated DSP processor solution that uses a specialized microprocessor whose architecture is optimized for measuring, filtering and converting analog signals to digital form. These microprocessors typically have the following characteristics:

- Do not contain flash to hold application code and must be loaded from an external source
- Require additional external components for input / output
- Perform integer mathematics faster than a standard microcontroller
- Nearly twice the cost of a comparable microcontroller

Dedicated DSP applications will often also still have a microcontroller to handle the device control application which can increase the solution cost and potentially the time to market.

"...developers don't need to use a separate, dedicated DSP processor to convert their analog circuits into software."

Thankfully, developers don't need to use a separate, dedicated DSP processor to convert their analog circuits into software. Developers working with Arm Cortex®-M processors can continue to use the Cortex-M programming model that they are most familiar with by selecting a processor that has built-in DSP instructions. These processors include the:

- Cortex-M4
- Cortex-M7
- Cortex-M33
- Cortex-M35P

Using a microcontroller with DSP extensions has several advantages over using a dedicated DSP processor. The benefits of using a microcontroller with DSP extensions include:

- Requiring a minimal number of external components
- A processor that already includes onboard peripherals, flash and memory
- No changes to the programming model, saving development time
- Improved performance in performing integer mathematics
- Decreasing the BOM costs

An important feature that developers will want to examine when making their processor decision for their application, is the processor performance based on the datatype that they need to process. Some microcontrollers with DSP extensions are designed to execute faster than others based on their target application. Figure 3 shows a relative comparison in performance between Cortex-M processors by specific datatype compared to a Cortex-M4 processor. This can be used by developers to determine how much performance they need out of their processor and assist in selecting the right processor for their application.

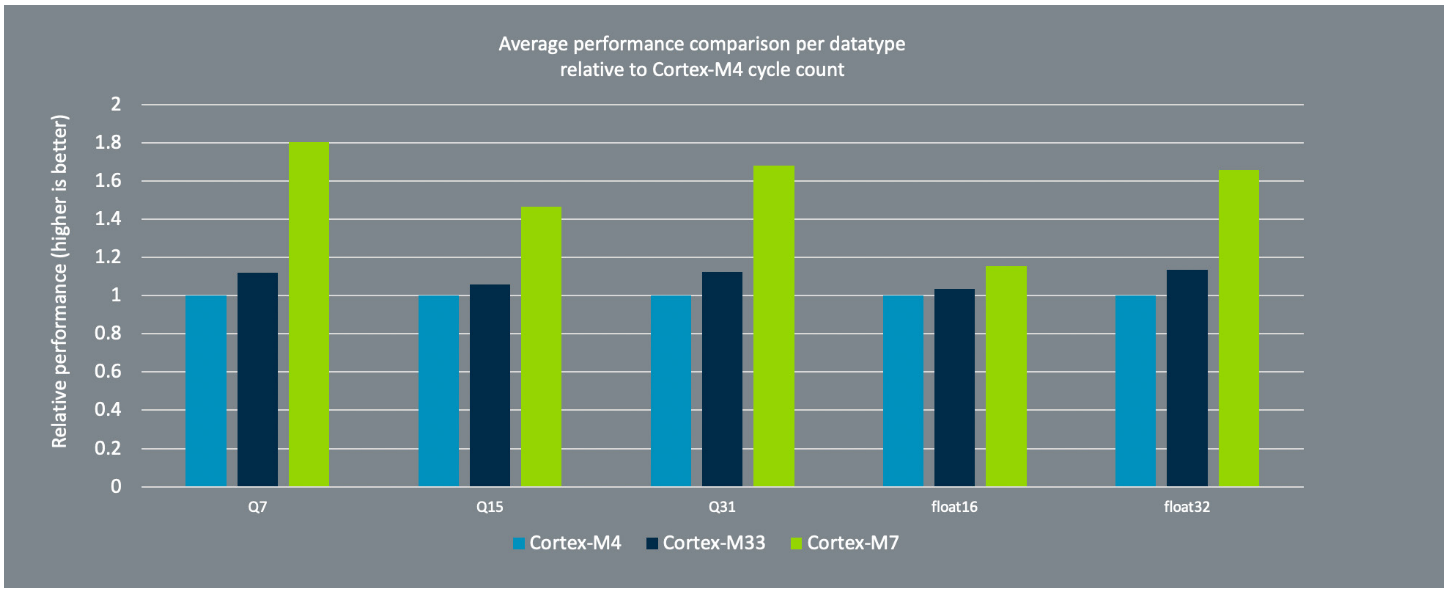


Figure 3: The DSP performance that can be obtained from the Cortex-M family will vary depending on the processor used and the data format. This graph shows the relative performance between Cortex-M processors with DSP extensions in relation to the Cortex-M4. The results were compiled from CMSIS-DSP kernels such as CFFT, FIR, RFFT, matrix multiply and vector dot products.

5. Capabilities and features of Arm Cortex-M with DSP extensions

The Arm Cortex-M processors that include DSP extensions offer a wide range of capabilities that help developers get their DSP applications up and running as quickly as possible. These capabilities can be broken up into two major categories; processor architecture features and software libraries.

When considering the processor architecture features, the Cortex-M can run DSP algorithms in either floating point mathematics or fixed point. The most efficient method is to use an onboard floating-point hardware accelerator, but for implementations where the silicon partner chose not to include the hardware accelerator, fixed-point mathematics can still be used to achieve good results. The speed at which the mathematics can be calculated is extremely important in DSP applications, especially when looking at deep learning and neural networks. These algorithms often use floating point math as part of the learning algorithm.

Beyond the ability to perform floating-point mathematics, the Cortex-M processor with DSP extensions include several processor instructions that are designed to speed up DSP algorithms.

These include:

- Saturating addition and subtraction instructions
- Multiply and Accumulate instructions (MAC)
- Single Instruction Multiple Data (SIMD) instructions

MAC instructions are used in many filter applications such as IIR and FIR filters. This dedicated instruction allows these algorithms to run much faster than if they were executed on a standard microcontroller. The SIMD instruction also will enable operations to be done on multiple 8-bit and 16-bit integer data during a single clock instruction. This helps the processor to get more done in a single instruction and results in the algorithm running faster.

The ability for the hardware to support DSP is crucial, but it is also important that there is a good ecosystem available for developers to use to speed up their design. There are resources available to Cortex-M developers, including the CMSIS-DSP library and Arm's DSP partners.

“The CMSIS-DSP library is a collection of over free 60 algorithms that make developing DSP software easier.”

The CMSIS-DSP library is a collection of over free 60 algorithms that make developing DSP software easier. The library contains functions for designing filters, calculating interpolations, performing complex mathematics and also performing transforms. A common use of the CMSIS-DSP library is to create IIR and FIR filters in addition to calculating a Fast Fourier Transform (FFT). The main algorithm categories that are available in CMSIS-DSP can be found in Figure 4.

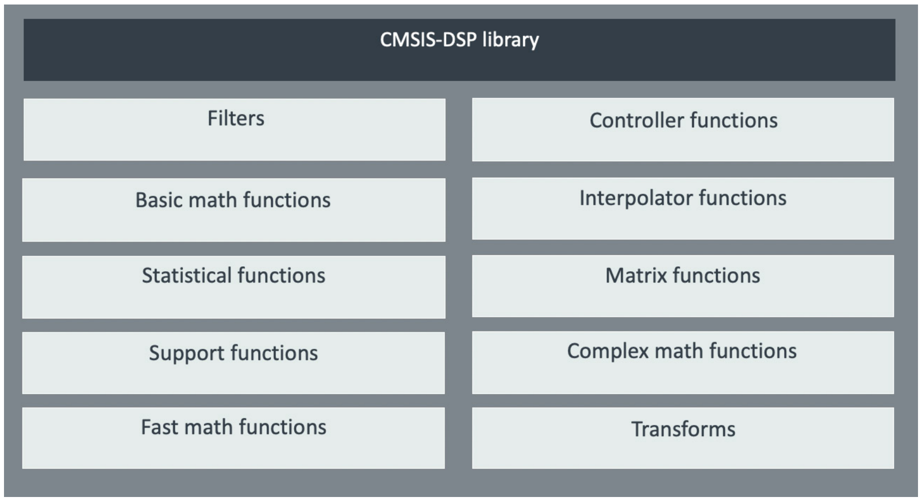


Figure 4: The CMSIS-DSP library is a free library containing over 60 algorithms that developers can use to speed up their DSP software development.

There are times when a development team doesn't have the skills or doesn't want to learn the skills that are necessary to implement a DSP algorithm. Sometimes there isn't enough time in a development schedule. In these cases, Arm's DSP partners who are experts in various DSP related fields can be used to speed up development.

6. Conclusion

Many factors within the embedded systems industry are driving the need to reduce BOM costs by converting analog circuits into DSP software. The conversion can now be done quickly and easily without the need for a dedicated DSP processor which is more costly than a DSP capable Cortex-M processor. The Arm Cortex-M processors and ecosystem have built the foundation that developers need to reduce their BOM costs while making the transition to using DSP software easy.

To learn more about DSP on the Arm Cortex-M processor, consider the following resources:

- Webinar: How to choose between analog hardware and digital signal processing
- Webinar: DSP software development masterclass with Arm and MathWorks
- Textbook: Digital Signal Processing using Arm Cortex-M based Microcontrollers

Additional resources to help you get started:

- Arm Cortex processors with signal processing
- Arm Cortex-M processors with signal processing
- CMSIS-DSP software library
- Arm's DSP ecosystem partners