An Independent Analysis of the:

MIPS Technologies MIPS32® M4K™ Synthesizable Processor Core

By the staff of

Berkeley Design Technology, Inc.

Overview

MIPS Technologies, Inc. is an Intellectual Property (IP) vendor offering a range of licensable 32- and 64-bit processor cores for use by SoC designers. These processor cores target applications ranging from deeply embedded, real-time control applications to high-performance embedded systems with demanding digital signal processing requirements. The M4K core is the smallest, lowest-power MIPS core, and is supported with a C and assembly language tool chain that includes application-specific software component libraries, an instruction set simulator, and in-system debug support. The M4K core is intended for deeply embedded control applications such as those found in wireless networking, automotive, and industrial control applications.

The M4K core is specifically designed to perform traditional control-oriented tasks in the embedded control applications for which it is intended. Historically, these applications do not include signal processing tasks. However, with increasing core clock speeds and architectural enhancements, processor cores competing with the M4K core, such as the ARM Cortex-M3, are increasingly finding some of their processing bandwidth being used for signal processing tasks. One area of interest, therefore, is how the M4K is likely to perform for such tasks. In this white paper, BDTI, an independent technology analysis company, assesses the capabilities and efficacy of the M4K core with a special emphasis on signal processing tasks.

Contents

Introduction

MIPS Technologies, Inc. is a provider of licensable, synthesizable 32- and 64-bit processor cores, and, with their recent acquisition of Chipidea®, a provider of licensable analog IP. Each member of the MIPS 32-bit processor core product line is compatible with the MIPS32® instruction set architecture (ISA) and targets a different class of applications. The MIPS32 ISA provides a standard instruction set architecture specification that is common across all 32-bit MIPS cores, allowing code written for one core to be reused with a different 32-bit MIPS core without modification. At the low end of the MIPS product line is the MIPS32® M4K® core. The M4K core has the smallest silicon footprint and lowest power consumption of all MIPS cores. It is targeted at cost-sensitive, deeply embedded control applications, such as those found in wireless networking, automotive applications, and industrial process control and instrumentation.

MIPS does not intend that the M4K be used for computationally-intensive digital signal processing (DSP) applications. Indeed, such applications are better served by larger, more powerful (and more costly) general purpose proces-

sors (GPPs) or DSP processors such as Texas Instruments' TMS320C54x and Analog Devices' ADSP-21xx. However, when compared with the 8- and 16-bit processors they are intended to replace in deeply-embedded control applications, the M4K core and its competitors, such as the ARM Cortex-M3 core, can provide significantly higher performance. Consequently, these cores are becoming increasingly attractive for simple DSP tasks, such as digital filtering and simple signal conditioning operations. While these DSP tasks may not be the primary function these cores perform in their target applications, they can augment the capabilities of the cores to provide additional features or to integrate certain system functions that would otherwise not be feasible. It is the efficacy of the M4K core in executing such DSP tasks that forms the basis for this paper.

The M4K core was introduced by MIPS in 2002. Like all MIPS cores, it has a number of build-time configuration options that allow it to be customized for different application requirements. Configuration options are applied prior to synthesis, and allow the designer to choose from a range of different implementations that trade off higher-performance for more compute-intensive applications against silicon area-efficiency to minimize die size and power consumption. The configuration options available include different levels of debug control and visibility, the inclusion of either high-performance or area-efficient multiply and divide hardware, different numbers of shadow register sets to improve interrupt performance, the MIPS16e™ Application Specific Extensions (ASE), and different types of instruction and data bus interfaces to the core. According to MIPS, one area-optimized implementation of the M4K core has a die area of 0.185 mm^2, typical power consumption of 0.066 mW/MHz, and a worst case maximum clock speed of 100 MHz, when implemented in a TSMC 130 nm G process. Also according to MIPS, one high-performance implementation, in the same target process, has a worst-case maximum clock frequency of 228 MHz, a die area of 0.64 mm^2 and typical power consumption of 0.066 mW/MHz. More details on clock speed, area, and power consumption are given in the section on Core Characteristics on page 5.

About BDTI

Berkeley Design Technology, Inc. (BDTI) is widely recognized for its long history as a credible source of independent analysis, evaluation, and benchmarking of processing engines and tools targeting embedded applications. In addition to detailed evaluation of the performance and architecture of DSP processors, BDTI's analysis activities have included significant focus on general-purpose microprocessors incorporating digital signal processing capabilities. This is exemplified by BDTI's groundbreaking 500-page technical report, DSP on General-Purpose Processors. Additionally, BDTI has completed numerous embedded software development projects targeting general-purpose processors, using a variety of operating systems including Linux and WindowsCE. For further information see www.BDTI.com.

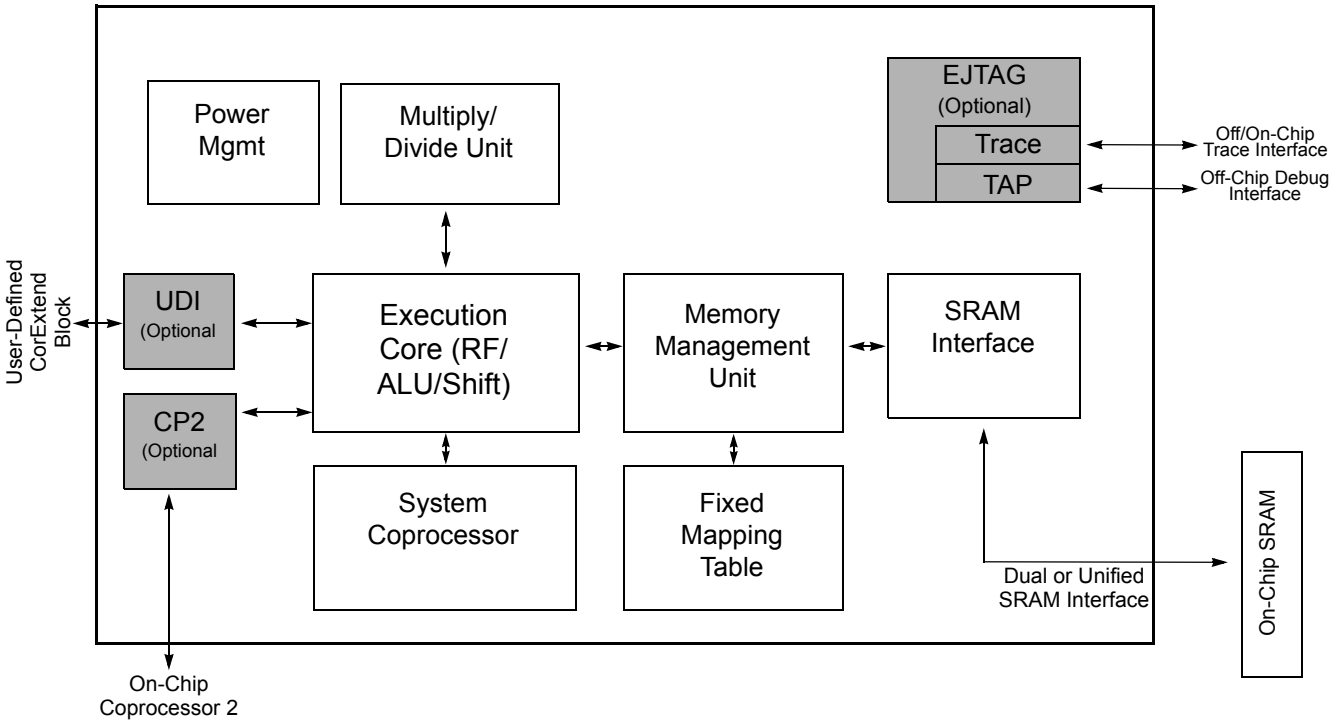

FIGURE 1. M4K™ Processor Core Block Diagram

Architecture

OVERVIEW

The M4K core is a 32-bit, integer RISC CPU that implements the MIPS32 Release 2 instruction set architecture. The data path comprises a 32-bit ALU, a shifter, a multiply/divide unit (MDU), and thirty-two 32-bit general purpose registers (GPRs). The M4K core uses a load/store architecture, where all ALU, shifter, and MDU operations execute on 32-bit data from, and return 32-bit data to, on-core registers. Load/store operations support 8-, 16-, and 32-bit data transfers to and from memory. A block diagram illustrating the key functional elements in the M4K core is shown in Figure 1.

While the use of a general-purpose processor for digital signal processing tasks may not achieve as high performance as a dedicated DSP processor, the M4K core architecture has features that enable it to provide reasonable performance for some DSP tasks. Among these is the inclusion of thirty-two 32-bit GPRs in the core. These GPRs can, for example, be used to store filter coefficients or operands that are shared among multiple calculations, reducing the memory transfer overhead and allowing higher computational throughput.

The key competing cores for the M4K core in cost-sensitive applications are the ARM ARM7TDMI and its more recent successor, the ARM Cortex-M3. In contrast to these cores, which have only sixteen 32-bit GPRs, the thirty two 32-bit GPRs in the M4K core is a distinct advantage. The greater number of GPRs enables the M4K core to store more coefficients, parameters, and data values on-core, allowing instruction slots that would otherwise be needed for memory transfer operations to be used to increase the throughput of computation operations.

MULTIPLY AND DIVIDE UNIT

Many DSP tasks are characterized by the need to process a large volume of real-time data using mathematically intensive operations. From a processor perspective, this means that the instruction flow is often dominated by multiply, multiply-accumulate (MAC), and add/subtract operations for data processing, and load/store operations for transferring operands and results to and from memory. In the M4K core, many mathematical operations are executed in a separate multiply and divide unit (MDU) that supports a number of different signed and unsigned multiply, multiply-accumulate (MAC), and divide instructions, all of which are part of the MIPS32 Release 2 ISA. As these instructions flow through the M4K core pipeline, they are transferred to the MDU for processing. Non-MDU instructions immediately following an MDU instruction continue to flow through the pipeline, without waiting for the MDU instructions to complete. The one exception to this is that any instruction following an MDU instruction that uses the result of that instruction will stall the M4K core pipeline until the result becomes available.

Two build-time configuration options are available for the MDU: a high-performance implementation and an area-efficient implementation. In the high-performance implementation, the MDU can complete one 32 × 16-bit (or one 16 × 16-bit) multiply or MAC operation in a single cycle. A 32 × 32-bit multiply or MAC completes in two cycles.

Most low-end DSP processors, such as TI's TMS320C54x or Analog Devices' ADSP-21xx, take significantly more than two cycles to complete a 32 × 32-bit multiply, highlighting the high-performance multiplication capabilities of the M4K core. Note that the ARM Cortex-M3 can perform a 32 × 32-bit multiply in a single cycle, while the older, ARM7TDMI cannot.

The alternative MDU configuration is the area-efficient implementation, based on a one-bit-per-clock iterative algorithm. In this configuration, all multiplies complete in 32 clock cycles, and all divides complete in 25 clock cycles, making it less attractive to chip designers targeting DSP-intensive tasks. As with the high-performance MDU, independent non-MDU instructions continue to flow through the M4K core pipeline while multi-cycle multiply, MAC, and divide operations are in progress.

Since the MDU can perform multiply and MAC operations in parallel with other ALU, shifter, and load/store operations, with careful scheduling of instructions the programmer may be able to execute data transfer operations in parallel with multi-cycle MDU operations and achieve higher computational throughput than would otherwise be possible. Multiply and MAC operations form the basis of many DSP algorithms, including convolution operations, FIR and IIR filters, and FFTs.

As a result of the difference in multiply and MAC throughput between the high-performance and area-efficient implementations of the MDU, ASIC designers contemplating the M4K core for DSP tasks will most likely opt for the high-performance MDU implementation.

Associated with multiply and MAC functions, many low-end DSPs typically include instruction support for rounding, saturation, and scaling operations. For example, the range of multiply instructions provided may include a variant that supports rounding the product of a 16 × 16-bit multiply to a 16-bit result. In many DSP tasks, the bulk of the processor cycles are often spent in tight inner-loops consisting of multiply and MAC operations. Since the M4K core does not provide hardware support for rounding, saturation and scaling operations, this will limit performance for the core for many DSP tasks. However, this is somewhat offset by the 32-bit data path in the M4K core (compared

to 16-bit data paths in typical low-end DSPs); this provides greater dynamic range and precision which may reduce the need for rounding, saturation and scaling operations. In addition, DSP library support for such operations can simplify the programming effort.

PIPELINE

The M4K core uses a five-stage pipeline that is fully interlocked: the pipeline will stall if an instruction requires the result of a previous operation that is not yet available. This frees the programmer (or compiler) from the need to consider pipeline hazards when implementing software. Bypass logic is included in the pipeline to ensure that the register-result of one instruction is available to the immediately following instruction, if required. For back-to-back instructions that first write and then read a core register, therefore, the pipeline will not stall. All ALU and shifter operations complete in one clock cycle. As discussed earlier, multiply and divide operations may take longer, depending on the operations and depending on the specific configuration of the MDU chosen by the designer.

Many DSP algorithms spend most of their time in computationally intensive inner loops. Such inner loops often comprise relatively few instructions meaning there are frequent branch operations associated with each iteration of the loop. To improve performance, most DSP processors include special purpose hardware for loops that eliminate or reduce the overhead associated with a branch. In the M4K core, like most general-purpose processors, no such loop hardware is included. Each branch takes two clock cycles to complete, which may add significant overhead to tight inner loops and impact performance for DSP tasks. If this is likely to be a key performance bottleneck, however, the user-defined instruction capability of the M4K core may be used to implement special purpose hardware to accelerate the loop, albeit with a significant degree of effort. Alternatively, loops may be unrolled to reduce the control overhead associated with branches. It should also be noted that although two clock cycles are required to complete a branch, with careful scheduling of assembly instruction, the programmer can use the branch delay slot immediately following the branch instruction to minimize the impact of branching.

The lack of hardware support to reduce loop-overhead in the M4K core is a disadvantage for DSP tasks since it will reduce performance for tight inner loops that form the heart of many such tasks. Many DSP processors, for example, include hardware support to eliminate this overhead. Note that neither the ARM Cortex-M3 nor the ARM7TDMI processor cores include hardware support for reducing loop overhead.

The five-stage pipeline in the M4K core is the shortest used in the MIPS 32-bit core product line. (In contrast, for example, the high-performance MIPS 74K core uses a 15-stage pipeline.) The use of a short pipeline in the M4K core is a limiting factor in the clock speed that can be achieved. A shorter pipeline, however, consumes less power and requires less silicon area, both of which are key requirements for cost-sensitive, deeply embedded applications. In addition, a shorter pipeline, together with its fully-interlocked design, is an easier target for efficient compiler code generation, and for hand-written assembly code.

For DSP tasks, the short and fully interlocked pipeline of the M4K core can reduce the effort required for optimized software design and accelerate software development efforts.

MEMORY SYSTEM

The memory system provided with the core consists of an SRAM-style (synchronous) external interface for instruction and data memory. No instruction or data memory, and no cache are included inside the core itself. For deeply embedded, real-time control applications, this arrangement can be a distinct advantage over cores that include on-core memory. In these applications, low cost is an important goal, and generally, memory requirements will be minimal. By using a relatively small SRAM (for example, less than 32 KByte), a designer can satisfy the memory requirements of the target application, while minimizing die size and, consequently, minimizing cost. The SRAM can be either on-chip (but off-core) or off-chip. In addition, using the smallest required memory helps to ensure that memory-related power consumption will be minimized.

For applications that have large memory requirements, the absence of an on-core cache in the M4K core may be a disadvantage. Including a large SRAM on-chip, but external to the M4K core, will be expensive and power-hungry, and may not be a viable solution. However, in such cases, a better choice of core would be the MIPS 4KE core, which does include an on-core cache memory and is otherwise very similar to the M4K core. For some M4K core applications, such as motor control or sequential process control, large memory is not a requirement and the lack of on-core memory in the M4K core will not be a concern. Many DSP tasks, however, involve the processing of a large-volume of data. For example, an FIR filter processing real-time data samples from a temperature sensor may have large data storage need, in order to read input data and write results. For processing of this nature, the 4KE core may be a better choice.

While the lack of on-core memory in the M4K core may be a disadvantage for tasks that have large memory requirements, the MIPS 4KE core may be an attractive alternative. The 4KE core is similar to the M4K core, but includes on-core memory.

As previously mentioned, a large number of real-time DSP tasks are characterized by the need for high computational throughput. This often involves mathematical calcu-

lations that operate on two operands. High-performance, therefore, is dependent upon the processor's ability to transfer operands and results to and from memory at a rate that can support the throughput of the mathematical calculations associated with that data. In low-end DSP processors, this is typically achieved by employing separate instruction and data busses. This allows one instruction and one data transfer to occur every cycle and provides a higher data memory transfer bandwidth than can be achieved with a single, shared bus for instructions and data. In high-performance DSP processors this idea is often extended by the use of two data memory busses in addition to a separate instruction bus, further improving the data transfer bandwidth and allowing more computational throughput for DSP tasks. The M4K core includes a build-time configuration option that implements separate instruction and data busses in the external memory interface. As with low-end DSPs, this improves DSP performance over a single bus by supporting higher computation throughput in the core. In contrast, for example, the ARM7TDMI includes only a single, unified instruction and data bus interface. ARM's successor to the ARM7TDMI, the Cortex-M3, includes separate busses for instructions and data. As with the M4K core, separate instruction and data busses in the Cortex-M3 core enable higher performance.

The use of separate instruction and data busses does have the disadvantage that more die area is required and that power consumption will be higher. For applications that don't need the improved data transfer bandwidth, the M4K core has a build-time configuration option that implements a single, unified instruction and data bus in the external memory interface. While this configuration reduces the data transfer bandwidth, it has the advantage that die-area and power consumption associated with the memory interface are minimized. The ARM Cortex-M3 core cannot be configured to use a single, unified instruction and data bus.

INTERRUPTS

Embedded control applications are often characterized by the need to handle multiple, independent control tasks with real-time response to system events. These requirements are most often handled with the aid of interrupt support provided in the CPU. In the M4K core, interrupt support is provided by three different build-time configuration options:

- An interrupt compatibility mode that is the same as that supported in the earlier Release 1 implementation of the MIPS32 architecture.
- A vectored interrupt mode.
- An external interrupt controller mode.

The vectored interrupt mode is the most powerful built-in configuration option for the core. In this configuration, the M4K core provides support for eight prioritized interrupt sources: six hardware interrupts and two software interrupts. A separate interrupt service routine can be associated with each interrupt. In addition, the M4K core can be configured to implement one, two, four, or eight sets of shadow GPRs. A set of shadow GPRs can be associated with each interrupt level. This eliminates the need for each interrupt service routine to explicitly save and restore context, and reduces the latency associated with interrupts. In embedded, real-time control applications, a faster response to system events means better overall performance. Consequently, the use of shadow GPRs is an important consideration for ASIC designers using the M4K core in embedded real-time control applications.

CORE CHARACTERISTICS

The different build-time options for the M4K core mean that a number of different configurations can be implemented, all of which will have different die size, clock frequency, and power characteristics. Table 1, below, shows these characteristics, as reported by MIPS, for two configurations. The first is a high-performance configuration that uses the high-performance MDU, separate instruction and data bus external memory interfaces, and extensive hardware debug support (which will be described in more detail, later). The second is an area-efficient implementation that uses the area-efficient MDU, no hardware debug support, and the unified instruction and data bus external memory interface. These implementations represent two extremes of the trade-off between performance and silicon area. It should be noted that the different build-time configuration options available for the M4K core provide varying degrees of trade-off between performance and silicon area. In generating the figures for Table 1, the optimization switches in synthesis and layout for the high-performance implementation targeted maximum clock speed, while those for the area-efficient implementation targeted minimum die size. The clock speed was obtained for worst-case process, voltage and temperature, while the power consumption was obtained for typical-case process, voltage, and temperature. In both cases, the target fabrication process was a 130 nm TSMC G process. The high-performance implementation used the TSMC-HP libraries and the area-efficient implementation used the ARM Metro libraries.

It should be noted that these implementations represent two extremes of the trade-off between performance and silicon area for the M4K core. The different build-time configuration options available for the M4K core enable a range of implementations between these two extremes that provide varying degrees of trade off between performance and silicon area.

From Table 1, it is evident that depending on the build-time options used for the M4K core and the optimization targets in synthesis and layout, clock frequency, die area, and power consumption can vary significantly. For ASIC

Core	M4K	M4K	Cortex-M3	Cortex-M3
Optimization Target	High-Performance	Area-Efficient	High-Performance	Area-Efficient
Max. Clock Speed	228 MHz	100 MHz	135 MHz	50 MHz
Die Area	0.64 mm^2	0.185 mm^2	0.74 mm^2	0.38 mm^2
Typical Power	0.214 mW/MHz	0.066 mW/MHz	0.165 mW/MHz	0.084 mW/MHz

TABLE 1. M4K® Core and Cortex-M3 Core Physical Characteristics for High-Performance and Area-Efficient Implementations (as Reported by MIPS and ARM). See the Text for Core Configurations, Target Libraries, and Target Processes

designers considering the M4K core for DSP applications, the high-performance implementation will be the most likely choice. This implementation provides single-cycle 32 × 16-bit, and two-cycle 32 × 32-bit multiplication and MAC support, separate instruction and data bus interfaces to memory, and operates at over 2X the clock frequency of the most area-efficient version of the core. For comparison, ARM claims that the speed-optimized ARM Cortex-M3 achieves a worst-case maximum clock frequency of 135 MHz in a 130 nm TSMC G process, using ARM Artisan SAGE-X libraries; around 59% that of the M4K core. With most instructions on both the M4K core and the Cortex-M3 completing in one or two clock cycles, this represents a significant speed advantage for the M4K core for DSP tasks. According to ARM, the speed-optimized Cortex-M3 has a die-size of 0.74 mm^2, without the Cortex-M3 Memory protection Unit and Embedded Trace Modules, and without most hardware debug support options. It does include the CM3Core, interrupt controller, and bus matrix. This implementation of the Cortex-M3 core is around 10% larger than the M4K core. Typical power consumption in the Cortex-M3, claimed by ARM for a speed-optimized configuration, is 0.165 mW/MHz; a reduction of around 25% over the M4K core. However, this does not present an advantage for ARM given that the area-efficient version of the M4K can be used at relatively high clock speeds and with substantially lower power consumption and smaller die size than the high-performance Cortex-M3 core implementation.

The area-efficient Cortex-M3 data given in Table 1, in contrast to the high-performance data, was obtained using ARM Metro libraries for a TSMC 130 nm G process.

The higher achievable clock speed for speed-optimized configurations of the M4K core suggest that it can achieve higher computational throughput for many DSP tasks when compared to a speed-optimized configuration of the ARM Cortex-M3 core. It is also interesting to note that while the power-efficiency of the high-performance M4K core implementation is less than that of the high-performance Cortex-M3 core implementation, the area-efficient implementation of the M4K core may present an attractive alternative for clock speeds up to around 100 MHz. For example, at 100 MHz, the high-performance Cortex-M3 core implementation will consume around 16 mW. The area-efficient M4K core implementation will consume around 6.6 mW, for the same clock speed. This corresponds to a 58% reduction in power compared to the Cortex-M3 core with a 75% reduction in die area.

Instruction Set

The M4K core, without the MIPS16e™ optional 16-bit instruction mode (discussed below), is a fixed instruction-length architecture, with each instruction constructed as a 32-bit word. The assembly language format will be very familiar to programmers who have used other general-purpose, load/store RISC processors.

Some notable instructions in the M4K core include the multiply instructions supported in the MDU, instructions for modifying and accessing bit-fields within 32-bit data words, and the count-leading-zeros (CLZ) and count-leading-ones (CLO) instructions, all of which are commonly used operations in many DSP tasks. For example, multiply operations commonly form the performance critical parts of filter and FFT tasks, and accessing bitfields is useful in extracting data words in variable word-length codes. In the high-performance implementation of the MDU the instructions for 32-bit multiply and multiply-accumulate complete in one or two clock cycles. While DSP processors may provide more powerful versions of these operations, including incorporating rounding and saturation support into them, the more limited versions in the M4K core do serve to extend the capabilities of a low-cost controller to some DSP tasks. In addition, as has already been described, the 32-bit data path in the M4K core can provide greater numeric precision and dynamic range, reducing the need for rounding and saturation operations, in contrast to most low-end DSP processors, which generally use a 16-bit data path.

The ARM7TDMI core does not support count-leading-zero and count-leading-one instructions, putting it at a disadvantage to the M4K core for those DSP tasks where those instructions can be used.

The MIPS16e capability is a build-time configuration option for the M4K core. When chosen, it provides for 16-bit instruction encoding in memory. When loaded into the M4K core, MIPS16e instructions are expanded to 32 bits for execution. While the 16-bit instruction encoding provides restricted functionality compared to 32-bit instruction

encoding, it has the advantage that code memory requirements are reduced significantly, since twice as many instructions can be stored in the same space. A second advantage of MIPS16e is that each instruction fetch reads two 16-bit instructions from memory, reducing the instruction fetch bandwidth requirement and, consequently, reducing power consumption.

In addition to the built-in instructions provided in the M4K core, the CorExtend build-time configuration option allows designers to add their own, User Defined Instructions (UDI) to the processor. UDIs are implemented by the designer in custom hardware, external to the core, but can access the core registers, and can complete in as few as one processor cycle. This capability allows designers to add functionality to the core to accelerate operations that present a performance bottleneck in the target application. ASIC designers using the M4K core for DSP tasks may find this a useful way to accelerate a few key operations in critical inner loops.

User Defined Instructions in the M4K core give the designer the ability to create custom instructions to accelerate performance-critical sections of code, such as those frequently found in the tight inner-loops of many DSP tasks. Neither the ARM Cortex-M3 nor the ARM7TDMI provide this capability.

Development and Debug Tools

MIPS provides tools and hardware features to enable software development and in-system application debug for the M4K core. The MIPS Software Toolkit is MIPS' premium development environment, consisting of the usual array of development tools an application developer might expect, and is common to all MIPS cores. It supports C and assembly language software development using a GNU-based compiler-assembler-linker tool chain. The Toolkit also includes the MIPS DSP libraries, and cycle-accurate simulation using the MIPSsim instruction set simulator.

The MIPS DSP libraries provide a wide range of functions, including math, delay line, filtering, correlation, FFT and DCT functions. The DSP libraries are written in C. Hand-optimized assembly versions of the C functions are available for the MIPS 4KE and 24K cores, but not specifically for the M4K. The DSP libraries provided by MIPS give programmers a jump-start in implementing DSP-related tasks.

In addition to software tools from MIPS, the M4K core is also supported by third-party software vendor Green Hills Software, which provides a C and C++ optimizing compiler, and an integrated development environment including software development management tools and a source-level debugger.

In addition to the software development environment, the M4K core includes build-time configuration options for different degrees of hardware debug support in the core, ranging from no debug support whatsoever, to support for complex hardware breakpoints and instruction and data tracing capabilities. These build-time configuration options provide the designer with some flexibility in making trade-offs between die size and power consumption on the one hand, and debug visibility and control on the other.

Basic hardware debug is enabled by choosing the enhanced JTAG (EJTAG) build-time configuration option. The EJTAG interface implements a proprietary extension to the IEEE 1149.1 standard that allows the designer to control debug events and single-stepping operations in the core. Three different build-time options for the EJTAG interface are available for hardware breakpoints that provide support for a different numbers and combination of breakpoint events. Breakpoint events can be used in different ways, such as initiating an exception handler for software-only debug handling, or for triggering instruction and data trace events.

For instruction and data tracing, MIPS provides build-time configuration options for either MIPS trace support, or a lightweight, iFlowtrace capability. MIPS trace allows for instruction and data tracing using either on-chip or off-chip trace memory. iFlowtrace only traces program flow, not data memory access, and provides a reduction in die-size over MIPS trace and also reduces trace memory requirements.

Additional debug support is provided by the First Silicon Solutions (FS2, a division of MIPS Technologies, Inc.) System Navigator product. System Navigator is a software tool that works with the trace hardware in the M4K core and with the GNU debugger provided in the MIPS SDE toolchain to allow users to capture and view program execution flow, load/store addresses and associated data. In addition to supporting the MIPS toolchain, System Navigator also supports third party tools such as Mentor Graphics' Edge, Green Hills Multi, and Viosoft Arriba.

Overall, for DSP task software development, the tools provided by MIPS are adequate. The DSP libraries will give programmers a good jump-start in coding their applications, and the in-system debug tools provided will accelerate the deployment of applications and products.

The inclusion of a cycle-accurate simulator is an important component of the software development tools provided for the M4K core, since it gives programmers the ability to carefully measure and tune the performance of tight inner-loops and sections of code that represent performance bottlenecks in many DSP tasks.

Strengths and Weaknesses

The M4K core is designed specifically for embedded control applications. Increasingly, DSP tasks are finding

their way into many such applications. While not specifically designed for DSP tasks, the M4K core does have strengths in this area when compared with competing 32-bit, cost-sensitive general-purpose processor cores. The most important of these strengths are:

- The M4K core uses a 32-bit data path which provides increased numeric precision and dynamic range over low-end 16-bit DSPs and which may reduce the need for saturation, rounding and scaling operations. The Cortex-M3 core, a key competitor for the M4K core also uses a 32-bit data path.
- The M4K core has thirty-two 32-bit registers which can be used to store DSP algorithm parameters, coefficients, and operands in order to reduce the load-store overhead of the processor and increase computational throughput. This is also a significant advantage over the ARM Cortex-M3, which has only 13 general-purpose registers.
- The M4K core can perform single cycle 32 × 16-bit multiply and MAC operations, which form the basis of many DSP tasks. This is comparable with the multiplier performance of low-end DSP processors. The ARM Cortex-M3 can perform 32 × 32-bit multiplies in a single cycle, but the older ARM7TDMI cannot.
- The M4K core has a build-time option for separate instruction and data bus memory interfaces. This allows an instruction and a 32-bit data word to be transferred every cycle, allowing better performance for most DSP tasks than can be achieved with a single, unified instruction and data bus. Where performance is less critical, the M4K core can be configured to use a single, unified data and instruction bus to reduce silicon die area and power consumption. While the older ARM7TDMI core used only a single, unified instruction and data bus, putting it at a significant performance disadvantage compared to the M4K core, the newer ARM Cortex-M3, like the M4K core, has separate instruction and data busses. Unlike the M4K core, however, the Cortex-M3 cannot be configured to use a single, unified data bus for reduced silicon die area and reduced power consumption.
- The count-leading-zeros and count-leading-ones instructions in the M4K core are useful for many DSP tasks, particularly in the absence of any other support for normalization. The ARM Cortex-M3 does not support these instructions.

The most important weaknesses of the M4K core for DSP tasks are:

- The M4K core has limited hardware support for rounding, saturation, and scaling operations. The ARM Cortex-M3 also has limited support for these operations. These operations are commonly supported in low-end DSPs and in many cases can be performed in the same cycle as multiply and MAC instructions giving these DSPs a significant performance advantage.
- The M4K core has no hardware support for zero-overhead loops. This can have a significant negative impact on performance for tight inner loops that form the basis of many DSP algorithms. The ARM Cortex-M3 also has no support for zero-overhead looping.
- The absence of on-core memory may be a disadvantage for the M4K core for those tasks that require the storage of large amounts of data. However, for such applications, the 4KE core, with on-core memory, may be a better choice.

Conclusions

The MIPS M4K licensable, synthesizable processor core is targeted at cost- and power-sensitive deeply embedded control applications. Increasingly, DSP tasks are finding their way into these applications. These tasks can supplement control-oriented tasks to provide flexibility and add application features that would otherwise not be feasible.

The M4K core, while not designed specifically for DSP tasks does have some strengths for processing such tasks. The most important of these includes the ability to perform a 16 × 16-bit multiplication in as little as one processor clock cycle, thirty two 32-bit general purpose registers for on-core coefficient, operand, and data value storage, and the ability to configure the core with separate instruction and data bus interfaces for increased instruction and data transfer bandwidth.

When compared with one of its leading competitors, the ARM Cortex-M3 core, the M4K core has a number of distinct advantages. These include higher operating clock speeds, smaller die area, and lower power consumption for many configurations and application scenarios. In addition to compeitive physical characteristics, the M4K core has thirty-two 32-bit general purpose registers giving it a significant advantage for many DSP tasks. The Cortex-M3 core-has only thirteen 32-bit general purpose registers . The one significant advantage that the Cortex-M3 has over the M4K core is that it can perform a 32 × 32-bit multiplication in a single clock cycle.

While the M4K core is not intended for compute-intensive DSP applications (anyone looking for a processor for such applications should look to dedicated DSP processors), it does present an attractive choice for deeply-embedded control applications where some simple signal processing capabilities would be an advantage.