An Introduction to MIPS32® microAptiv™ Processor Cores - The Convergence of MCU/MPU and DSP

Document Number: MD00928
Revision 01.00
May 9, 2012

MIPS Technologies, Inc.
955 East Arques Avenue
Sunnyvale, CA 94085-4521

Overview

The new microAptiv™ family of processor cores, part of the new Aptiv™ Generation of cores from MIPS Technologies, are highly-efficient, compact, real-time embedded MIPS32® Release 3 processor core solutions for microcontrollers (MCU) and entry-level embedded market applications.

The microAptiv cores build on the MIPS32® M14K™ core family, leveraging the highly-efficiency microMIPS™ code size reduction Instruction Set Architecture (ISA) and high-performance 5-stage pipeline design.

The microAptiv cores add DSP functionality and SIMD (Single Instruction Multiple Data) capability, enabling the convergence of RISC/MCU technology and DSP—delivering cost-effective solutions for a wide range of embedded applications that require more signal processing performance.

As with the M14K family of cores, the microAptiv cores are available in two versions:

- microAptiv MCU core - a cache-less design with integrated high performance, low latency SRAM interface
- microAptiv MPU core - with integrated cache controller, Memory Management Unit (MMU) and standard AMBA® AHB™ Bus Interface Unit

In addition, the microAptiv MCU core provides support for security with the inclusion of a Memory Protection Unit and a secure debug feature that prevents access to the core from the debug port.

MIPS Technologies continues its commitment to reducing SoC design and product cost by providing an option for a 2-wire cJTAG interface (IEEE standard 1147.9), enabling the use of low-cost debug probes.

1 Convergence of MCU/MPU and DSP

In today's world, we come into contact with MCU systems several hundred times a day. At home, on our travels, at our place of work and beyond, we use and rely upon MCUs for practically everything we do. MCUs control our household appliances, manage the convenience functions, engine and entertainment in our cars, and run countless number of systems in our offices and factories.

Traditionally, MCUs have been used to control applications in the analog domain, where real-time, deterministic performance with 'just enough' data processing have been key requirements. The more advanced features in today's embedded systems, which incorporate high-performance communications, multimedia, control and sensor technologies, require MCUs to operate more effectively in the digital domain by providing more complex data processing capabilities.

This has resulted in the concept of a Digital Signal Controller (DSC), a term first introduced by Microchip Technology, which integrates the interrupt-driven control functions of an MCU with the high performance math execution strengths of a DSP, combining the best of both worlds in a single package.

Some of the benefits of a unified MCU/DSP system include:

- Reduced cost of ownership: eliminates the need for a dedicated DSP. Further cost reductions are made possible by the DSP being tightly integrating into the CPU logic, enabling the sharing of core logic and on-chip resources.

- Faster time to market: reduces design and development time, since developers only need to use a single toolchain to design and debug both MCU and DSP code.

- Reduces software complexity: reduces the learning requirements to a single architecture. The design can be implemented in a single RTOS environment, minimizing development and support of the application software by having a single code base.

- Optimizes performance: provides a degree of parallelization and reduces latencies between operations the MCU/MPU and DSP perform.

- Re-use of IP: provides a platform for re-use in broader range of existing and future systems.

Of course, these benefits don't come without challenges. Challenges in developing such a solution include:

- The combined MCU/DSP solution must have enough performance to deliver high throughput, single cycle, DSP execution at the maximum rated operating frequency of the core (i.e. no performance should be lost).

- The real-time performance of the MCU logic should not be compromised.

- The range of supported MAC (Multiply and Accumulate) and data processing instructions should be comparable to those available on dedicated DSPs.

- Adding DSP functionality should not significantly increase the core size and power.

The microAptiv cores incorporate techniques that minimize latency in the critical timing path of the Arithmetic Logic Unit (ALU). Logic dedicated to executing DSP instructions is added in parallel with the existing ALU blocks. This not only helps to maintain maximum core operating speed, but also allows simultaneous execution of ALU and DSP operations and back-to-back ALU/DSP instructions to complete with no additional cycle latency. The additional core area is minimized by reusing the ALU's 32-bit adder and shifter, and by redesigning/re-using the Multiplier Array to implement most of the DSP multiplies logic without affecting the timing and execution of existing multiply instructions.

Furthermore, the microAptiv implementation includes DSP MACs which can be executed consecutively at optimal repeat rates. A table of all ALU and DSP instructions and their corresponding throughput cycle rates is shown later in this paper.

2 microAptiv™ Core Product Features

The microAptiv family consists of two processor cores: the microAptiv MCU core and the microAptiv MPU core. Both are supersets of their respective M14K family core versions, with the microAptiv MCU core (Figure 1) building on the M14K microcontroller design, and the microAptiv MPU core (Figure 2) building on the M14Kc embedded processor core.

Figure 1 microAptiv™ MCU core

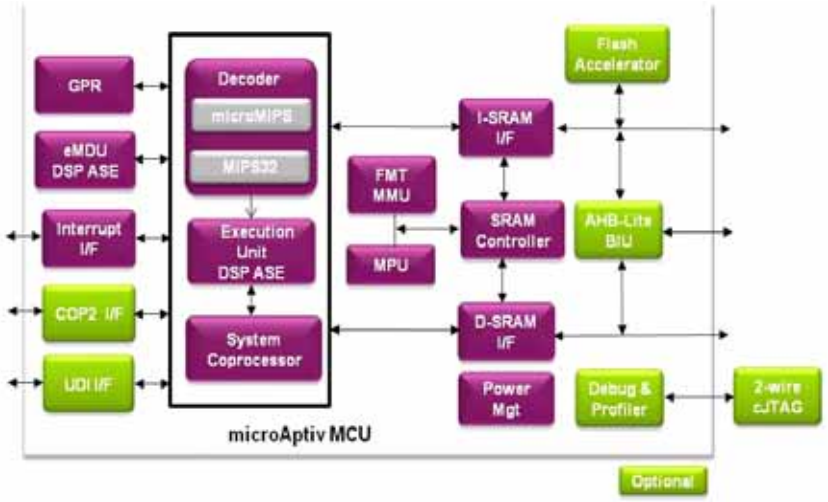
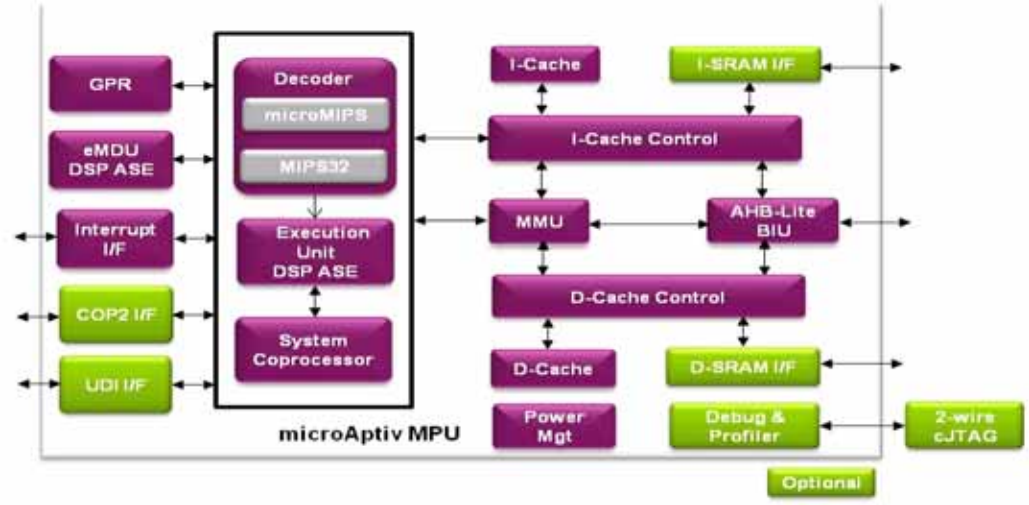


Figure 2 microAptiv™ MPU core



Both microAptiv cores have several features in common which are present in the base architectures, namely:

- 5-stage pipeline architecture
- High-performance execution unit, capable of delivering 1.57 DMIPS/MHz and 3.44 CoreMark/MHz performance

- Support for both a MIPS32 and microMIPS instruction decoder, with an option to execute in microMIPS-only mode
- MIPS Release 3 Architecture
- MCU ASE (Application Specific Extension) which reduces interrupt latency and extends the interrupt delivery mechanism over previous generation cores, and two memory-to-memory atomic instructions for semaphore manipulation which is commonly used in MCU application
- Up to 16 sets of 32x32- bit General Purpose Registers (GPRs)
- Fixed Mapping Translation Unit Memory Management Unit (FMT MMU)
- Multiply Divide Unit, including a hardware multiplier array for MAC instructions
- AHB-Lite BIU
- EJTAG debug and trace capabilities, including support for iFlowtrace™, Performance Counters (PCs) and PC address and data sampling
- Support for 2-wire cJTAG debug interface
- Secure debug feature which prevents streaming instructions through the EJTAG port
- Power management

In addition, the microAptiv MCU core includes a Memory Protection Unit that provides access control for up to 16 programmable memory segments.

The microAptiv MCU core targets microcontroller systems without on-demand paging, which tend to be designed with both Flash and SRAM memory. To maximize memory throughout, the microAptiv core implements a high-speed SRAM controller, and a Flash Accelerator that reduces access times to code that is resident in flash memory.

The microAptiv MPU core integrates additional features of programmable cache controller and Translation Looka-side Buffer (TLB) MMU to facilitate embedded processor designs executing operating systems which manage virtual memory with on-demand paging, such as Linux and Android™.

The main enhancement of the microAptiv cores is the ability to accelerate complex signal processing functions with the addition of DSP ASE release 2 (r2), which until now was only available on higher-end MIPS cores. Included in the DSP ASE are a 32-bit SIMD engine and enhanced 32x32-bit MDU that accelerate the performance of a large set of Multiply/MAC and other signal processing instructions.

3 Digital Signal Processing

Before we get into the details of the DSP functionality in the microAptiv cores, let’s review a couple of key aspects of digital signal processing.

3.1 Type of Signal

In the digital signal processing domain, a signal is a set of data samples representing the change in time or space of a digitized analog waveform or digital state of a system. Typical examples of signals are the sound waves output from a microphone, positional data of a motor or the carrier frequency of a mobile radio.

Signal data is provided as a function of the sampling rate (the rate at which the signal measurement is taken) which is determined by the frequency range, and the resolution (the number of bits used to determine the signal value) which is set by the dynamic range and quality of the application. For instance, voice is measured at an 8/16-bit sample rate over a 50-100db dynamic range and across a 0-8KHz frequency range. Similarly, high quality audio (MP3 for example) is generated at 16/24-bit resolution at samples rates typically at 48 KHz frequency, up to 150db dynamic range.

3.2 Data Types

DSP algorithms are usually represented in fixed-point fractional format, between +1 and -1. The binary point separating the integer and fractional parts of the number is located immediately after the sign bit. The more common data types do not include any integer bits:

- Q15 – sign bit, 15 fractional bits
- Q31 – sign-bit, 31 fractional bits

3.3 MAC Operations

Multiply-and-Accumulate (MAC) are the most commonly executed DSP operations, being used extensively in DSP algorithms, especially common in filter and transform functions. A typical example of the use of a MAC operation is a digital filter that multiplies signal samples by filter coefficients and sums up (accumulates) the products to obtain a single output sample, called a Dot-Product.

3.4 Saturation and Rounding

Saturation and rounding operations are regularly part of DSP algorithms, used primarily to avoid overflow and underflow conditions. As an example of the benefit of saturation, consider the following application in a typical MCU system: In an 8-bit graphic controller, a black pixel has a value of 0, while white is represented with a maximum value of 255. Imagine the effect of increasing the brightness by 25% of a near while pixel that has a value of 220, resulting in 220 x 1.25 = 275. This value cannot be represented in 8 bits and so would be wrapped around to produce 275 – 256 = 19, which would be displayed as nearly black. Using the saturation technique avoid this problem by clipping the result to the maximum value of 255.

Rounding is used when the precision of multiply operations or the sequences of MAC operations is required to be reduced. Rounding improves the stability and precision of the calculation in comparison to using truncation alone.

Normally multiplying two Q15 numbers produces a Q31 result. But if the result needs to be stored as a 16-bit value, then rounding is required. Consider the example: 0.5 x 0.5 = 0.25. We want to reduce the precision to one decimal point, but do we choose 0.2 or 0.3? Using truncation will result in the value being stored as 2.0. In fact, any number between 0.20 to 0.29 (0.20, 0.21, 0.22, ….0.29) would be stored as 0.2, which would result in the algorithm that uses this calculation to be unstable. Adding a rounding value of 1/2 the least-significant digit in the above sequence, then truncating, gives a more precise result – values 0.20 to 0.24 would truncate to 0.2, while values 0.25 to 0.29 round up to 0.3.

The MIPS32 DSP ASE supports signed/unsigned and fractional data types, 8/16/32-bit MAC operations, and includes both saturation and rounding options in a wide range of instructions.

4 MIPS32 DSP ASE

The MIPS32 DSP ASE is a combined hardware and software solution designed to improve the performance of DSP applications. It is implemented by reusing as much circuitry that is already present in the core as possible, with minimal impact on area and operating frequency.

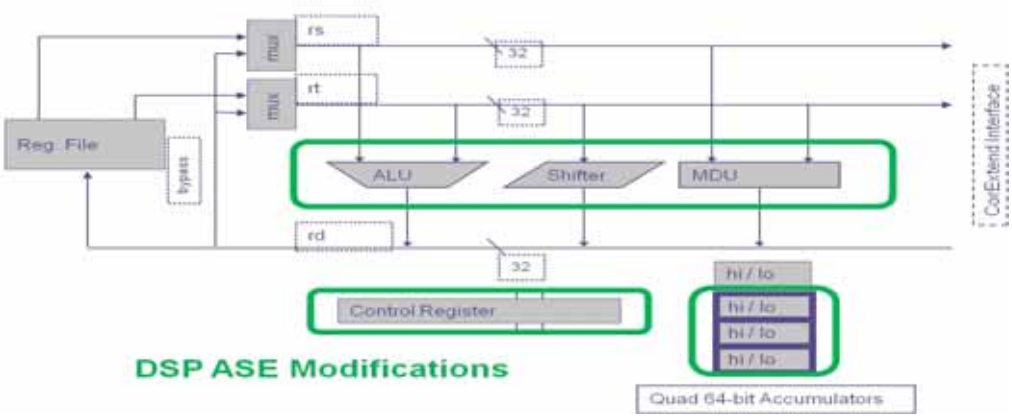
Three additional 64-bit accumulators (for a total of four) have been added to improve the performance of MAC-intensive algorithms.

4.1 Pipelines

The microAptiv cores implement two independent 5-stage pipelines: an ALU pipeline that implements the standard load/store, computational, branch instructions and an MDU pipeline. The two pipelines operate in parallel, enabling non-dependent ALU instructions to execute while the MDU is busy, increasing the overall performance of the microAptiv cores for either DSP or MCU-specific applications. Any stall in the ALU pipeline does not affect the execution of the instruction underway in the MDU pipeline. The DSP ASE uses both the ALU and MDU pipelines to execute its instructions.

In addition to integer multiply and divide operations, the MDU pipeline also executes DSP ASE multiplication-type instructions and instructions that access the accumulators. In addition, appropriate forwarding, where the result of a previous instruction is sent directly to the current one bypassing the register file update, is provided inside the ALU pipeline and between the ALU and MDU pipelines. Figure 3 shows how the DSP ASE logic fits into the base core architecture.

Figure 3 DSP ASE Additions



4.2 Instructions

The DSP ASE implements over 150 instructions, including 70 SIMD and 38 Multiply/MAC instructions, operating on 8/16/32-bit signed/unsigned integer and fractional data types with a single cycle throughput. The types of supported instructions include a range of arithmetic with saturation and rounding option, data packing/unpacking typi-

cally used in scaling functions, compare/pick, load and accumulate operations. In addition to the commonly found operations, the DSPASE also includes some advanced features that improve performance without requiring a complex implementation. An example is a variable bit-extract method that efficiently extracts bits from an incoming stream. Another is a feature that efficiently processes complex numbers. The ASE also includes a novel and efficient way to support virtual circular buffers.

The complement of Multiply/MAC instructions are enabled by an integrated 32x32 multiplier array that executes 32x32, 16x16, dual 16x16, dual 16x8 and dual 8x8 multiply operations. The result of the Multiply/MAC instruction can be written to either the GPRs or Accumulators.

The DSP ASE implements a register-based SIMD feature where single instructions can execute multiple operations on multiple data operands. The range of SIMD instructions support add, subtract, shift and multiply of 8, 16 and 32-bit data types, and are able to operate on up to four operands simultaneously. SIMD data is packed into a GPR, which can contain one or more operands of each size up to four 8-bit or two 16-bit values.

The instruction mnemonic indicates the supported data types:

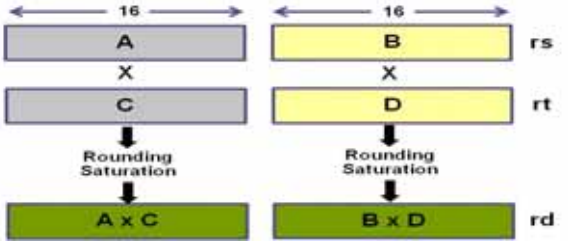
– W = 32-bit ‘Word’

– PH = ‘Paired Halfword’ 2x16-bit

– QB = ‘Quad Byte’ 4x8-bit

An example of a single cycle SIMD instruction is MULQ_RS.PH rd, rs, rt. This is a multiply instruction (MUL) that multiplies two vector elements of fractional (Q) 16 bit data (PH) with rounding and saturation (_RS). Each source register supplies two data elements and the two results, after truncation, are written into the destination register as shown in Figure 4.

Figure 4 Example of SIMD Instruction



4.3 Instruction Repeat Rates

Figure 5 shows the repeat rates (throughput) for combinations of sequences between two arithmetic, integer multiply and DSP/SIMD multiply instructions, with and without data dependency.

Figure 5 DSP Instruction Throughput

	Instruction Sequence				Repeat Rate	
	1st Instruction		2nd Instruction		Without Data Dependency	With Data Dependency
	Instruction	Destination	Instruction	Destination		
Integer Instructions	Arithmetic	GPR	Multiply**	GPR or ACC	1	1
	Integer Multiply	GPR			1	4
		ACC			1	1
DSP ASE Instuctions	Arithmetic	GPR			1	1
	SIMD Multiply, MAC	GPR			1	4
	Dot Products	ACC			1	1

Multiply** includes SIMD multiply and MAC instructions where the destination is a GPR, and a Dot Product multiply instruction that stores the result in one of the 4 Accumulators. Data dependency means that the second instruction depends on data from the first instruction, or that uses the same destination.

4.4 Software Utilization and Optimization

There are several techniques that can be implemented when using the DSP ASE to benefit increased performance and reduced software development time when writing DSP code for the microAptiv core.

The GNU Compiler, supplied in support of the microAptiv core, enables the use of the following options to develop code:

- Assembly: hand-coding in assembly language produces code with the highest performance, and is particularly beneficial when used in the most frequently used routines
- asm macros that produce DSP instructions directly from C code
- Intrinsics: the use of a wide range of intrinsics supporting both integer and fractional data types and for use of the 64-bit accumulators; the use of intrinsics over the assembly options can result in more efficient code execution, as the compiler has knowledge of the pipeline latency and is therefore able to schedule the execution of DSP instructions more optimally
- Support for fixed point data types and operators

The DSP performance of the microAptiv core can be enhanced by better understanding the benefits that the architecture provides, and by implementing optimizations that are specific to the DSP ASE. Namely:

- Select the most appropriate algorithm for the application. For instance, in an FFT design, radix-4 provides higher performance than radix-2 at the expense of a reduced set of supported sample sizes.
- Select the smallest possible data type. This allows more parallelism in executing microAptiv SIMD operations and will yield a smaller cycle count and higher performance.
- Implement loop unrolling to reduce the housekeeping overhead associated with pointer update management and branching. By unrolling the loop, the code size would increase marginally, but the cycle count could be significantly reduced.

- More efficient scheduling of instructions by utilizing software pipelining, distributing sequences performed sequentially in a single iteration into stages across several iterations. This eliminates data dependencies between instructions in a single iteration and reduces stalls.

- Reduce the number of memory accesses (typically high when implementing filter algorithms) by storing the data and coefficients required in the calculation once from memory into the microAptiv core GPRs.

- Take advantage of the wide range of instruction types available in the DSP ASE. For example, data packing/unpacking is a common procedure used in audio and video compression algorithms. Recognizing the importance of this, the DSP ASE includes instructions to accelerate bitstream processing.

5 DSP Library

MIPS Technologies provides a DSP Library included with the microAptiv core which consists of a number of algorithms that perform Fast Fourier Transforms (FFT), common filter calculations, vector operations and even H.264 video compression operations. These specific algorithms were developed as they are common functions used in the development of a wide range of MCU/DSP applications across a broad spectrum of embedded segments, including industrial networks, motor control, smart meter management, security cryptography, video decoding and voice compression.

Several of these algorithms are developed using the optimizing techniques outlined in the previous section. Critical performance code is implemented in assembly code; other less-frequently used, non-critical routines are implemented in C. The microAptiv version DSP Library routines uses DSP ASE r2 instructions coded in microMIPS format as much as possible, and are scheduled to operate efficiently within the 5-stage pipeline architecture. The functions implemented in the DSP Library are shown in Table 1.

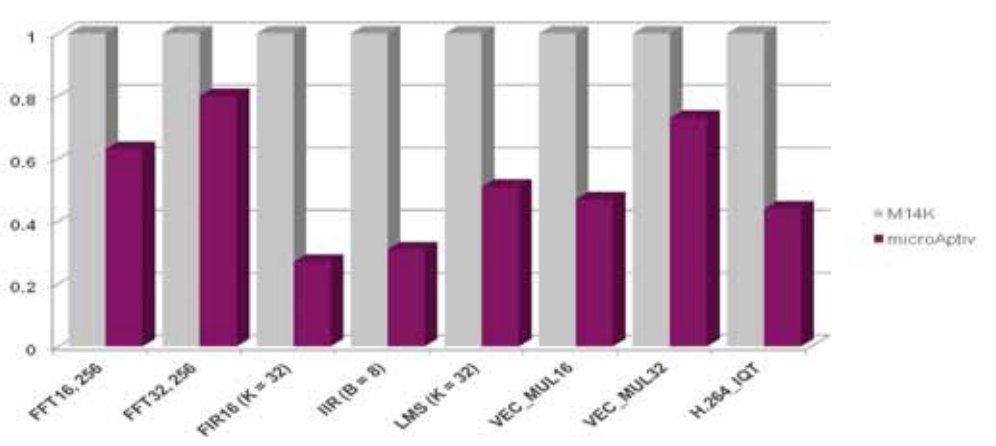
Category	Function	Description
Transform	FFT16	Compute complex FFT of a Q15 vector sample
	FFT32	Compute complex FFT of a Q31 vector sample
Filters	IIR16	Single sample IIR filter
	FIR16	Applies a block FIR filter to a Q15 vector
	LMS16	Single sample LMS filter
Vector Math	vec_abs16/32	Compute the absolute value of each Q15/Q31 vector element
	vec_add16/32	Add the corresponding elements of two Q15/Q31 vectors
	vec_addc16/32	Add a constant to all elements of a Q15/Q31 vector
	vec_dotp16/32	Compute dot product of two Q15/Q31 vectors
	vec_mul16/32	Multiply the corresponding elements of two Q15/Q31 vectors
	vec_mulc16/32	Multiply all elements of a Q15/Q31 vector by a constant
	vec_sub16/32	Subtract the corresponding elements of two Q15/Q31 vectors
	vec_sum_squares 16/32	Calculate the sum of squares of elements of a Q15/Q31 vector
Video	H264_iqt	Inverse quantization and transform for H264 decoding
	H264_luma	Computes 1/4 pixel motion compensation for luma pixels in H264 decoding

Table 1 DSP Library Functions

5.1 Results

The chart of Figure 6 illustrates the cycle counts of a sample of the DSP Library functions, comparing the cycle counts of the microAptiv core against the M14K core. The microAptiv core demonstrates a 1.5 – 3.7x decrease in the number of cycles required to execute the function compared to the non-DSP enabled M14K core.

Figure 6 DSP Cycle Count Comparison



6 Comparison with Cortex-M4

Both the microAptiv cores and Cortex-M4 have been developed to address the digital signal control market, integrating MCU and DSP functionality. However, the microAptiv cores provide a richer set of MCU application-specific features, and higher performance, greater range of DSP capabilities than is available with the Cortex M4. A list of the major differences and enhancements offered by the microAptiv cores are listed in Table 2.

:

FEATURE	microAptiv	Cortex M4
Pipeline Stages	5	3
Cache/MMU version	Y	N
ISA	MIPS32 and/or microMIPS	Thumb2
Total instructions	300	155
GPRs	32	16
Closely coupled memory support	Y	N
Interrupt latency	10 cycles	12 cycles
Instruction-only trace	Y	N
Fast Debug Channel	Y	N
DSP FEATURE	**microAptiv**	**Cortex M4**
Total DSP Instructions	159	80
SIMD Instructions	70	38
Multiply/MAC instructions	38	29
Dedicated DSP/MDU unit	Y	N
Accumulators	Y (4)	N
16x8, dual 8x8 Multiply/MAC	Y	N
Shift Instructions	Y	N
Compare/Pick Instructions	Y	N

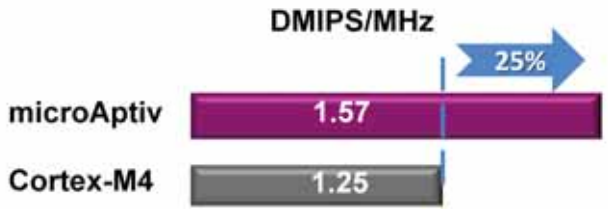
Table 2 Feature Comparison: microAptiv Core and Cortex-M4

The Cortex-M4 is basically the Cortex-M3 with the addition of simple, limited DSP functionality. The microAptiv cores, on the other hand, are supersets of the M14K/c cores with enhancements to include DSP extensions that exist on MIPS' high performance solutions, a microMIPS-only operating mode and additional debug features.

6.1 Processor Performance

The microAptiv core deliver 25% more performance than Cortex-M4 on the Dhrystone benchmark. For CoreMark, the CoreMark/MHz numbers for microAptiv and CoreMark-M4 are the same. However, the microAptiv result is achieved using a free gcc compiler, whilst that of Cortrex-M4 is from using a pay-for custom compiler as shown in Figure 7.

Figure 7 Dhrystone and CoreMark Comparisons



6.2 Architecture

Differences exist between the MIPS and ARM architectures that benefit the microAptiv cores and contribute to their higher performance, as well as ease of software development relative to Cortex-M4:

- The microAptiv cores are available in two versions: the microAptiv MCU core developed with real-time, reduced latency, low area and low power features required of MCU designs, and the microAptiv MPU core with a high performance ALU and MMU for supporting virtual memory embedded processor designs.
- microAptiv cores include an option for shadow registers that enable context switching between processes and interrupts.
- The MIPS architecture has mostly single operation instructions, while ARM, for many instructions, executes multiple operations before writing the result to the GPR.
- MIPS has more simpler addressing modes.
- The microAptiv cores implement an efficient branching design, which obviates the need for complex branch prediction.
- The microAptiv cores support several architecture extensions not available with the Cortex-M4, in particular co-processor and CorExtend® User Defined extension interfaces.

6.3 DSP Implementation

The DSP functionality is implemented differently and more effectively in the microAptiv cores compared to that in the Cortex-M4.

microAptiv DSP extensions are designed to operate with their own dedicated pipeline, separate and in parallel to that of the core ALU. The microAptiv DSP shares some of the multiplier and register file logic with the ALU execution unit.

The Cortex-M4 uses a shared pipeline design between the DSP and ALU functions.

The microAptiv cores include up to four 64-bit accumulators in addition to GPRs to store the results of the DSP operations. The Cortex-M4 does not implement any accumulators, using shared GPRs (and only up to 16) to store the results of either the DSP or ALU operations.

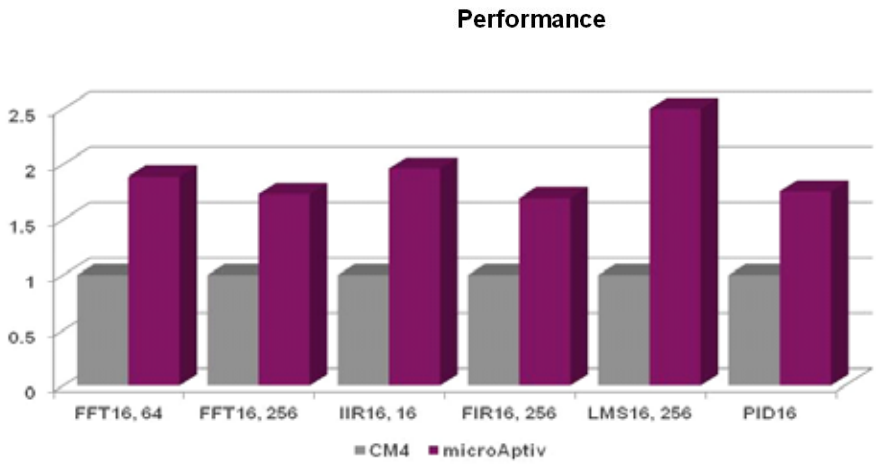
The microAptiv cores implement twice the number of DSP instructions compared to the Cortex-M4, with a richer set of functions and more SIMD and Multiply/MAC instructions. In addition, microAptiv has more flexible scaling/shifting operations.

6.4 DSP Performance

ARM provides a software application layer for the Cortex-M4 called Cortex Microcontroller Software Interface Standard (CMSIS). Included in CMSIS are a set of commonly used DSP algorithms, similar to those provided in the MIPS DSP Library.

Differences show up in performance when benchmarking functions that are common to both the MIPS DSP Library and CMSIS. The Cortex-M4 results were obtained on a STM ST32FM4 Discovery board using the Keil MDK development tools. Results for the microAptiv core were measured on a MIPS SEAD-3 board using Mentor Sourcery CodeBench gcc compiler. Figure 8 illustrates the performance advantage that the microAptiv cores have compared to Cortex-M4 when executing FFT, IIR, FIR, LMS and PID functions:

Figure 8 DSP Performance Comparison, microAptiv Core and Cortex-M4



As an example, for FFT16, the microAptiv core delivers more than 1.7x the performance of the Cortex-M4, approximately 50% less cycles to execute.

7 Summary

The combination of real-time performance with comprehensive, high-throughput DSP functionality in the new microAptiv cores creates an ideal solution for systems applications requiring more signal processing performance within a minimum area and power budget.

Template: nW1.03, Built with tags: 2B