Enhancing the Computational Performance of the C2000™ Microcontroller Family

Kenneth W. Schachter *C2000 Technical Staff*

ABSTRACT

Engineers designing real-time control systems are constantly faced with the challenge of optimizing performance. These systems require minimal processing latency in order to meet the control loop performance specifications. At the heart of the control systems are math intensive algorithms which are used to calculate the control signals. Utilizing a microcontroller (MCU) that can quickly and efficiently execute mathematical operations is critical towards this objective. Ideally, this MCU would be able to execute the real-time control loops concurrently with the central processing unit (CPU) while it is performing other required tasks. This paper discusses five integrated on-chip hardware math enhancements that dramatically increase the performance of the MCU in many real-time applications. These math enhancements boost the CPU processing capabilities by utilizing extended instruction sets, additional registers, and hardware. When combining a high performance CPU with these advanced hardware enhancements, the fast and efficient processing power required for complex real-time control systems can be realized.

Contents

List of Figures

List of Tables

Trademarks

C2000 is a trademark of Texas Instruments.

1 Introduction

Real-time control systems require fast and efficient processing, with latency kept to a minimum in order to maintain stability and boost overall performance. In addition, the increasing sophistication of modern motor systems, power electronics, smart grid technology, robotics, and similar applications require the central processor to keep up with numerous tasks simultaneously.

The C2000 family of microcontrollers (MCUs) from Texas Instruments addresses these challenges with an array of integrated on-chip hardware math enhancements that dramatically increase the performance of the MCU in many real-time applications. The five key enhancements are:

- Floating-Point Unit (FPU)
- Control Law Accelerator (CLA)
- Trigonometric Math Unit (TMU)
- Fast Integer Division Unit (FINTDIV)
- Viterbi, Complex Math, and CRC Unit (VCU)

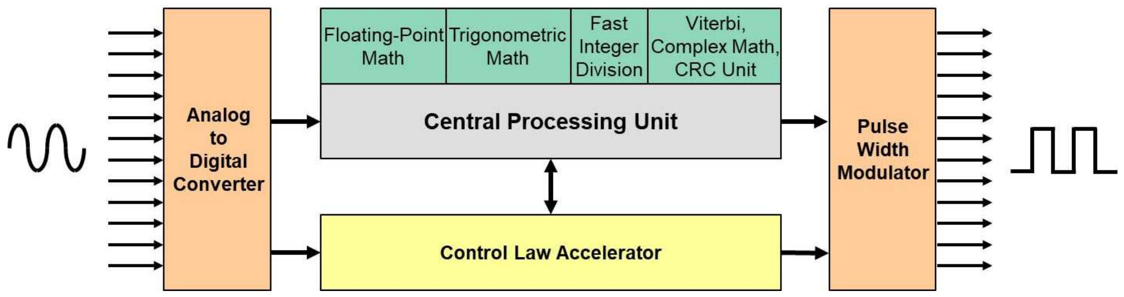

Figure 1. System Block Diagram with Math Enhancements

At the center of each C2000 MCU lies a fast fixed-point central processing unit (CPU) that on its own provides excellent 32-bit processing capabilities. The FPU provides seamless integration of floating-point hardware into the CPU. To augment this further, the CLA provides an independent floating-point CPU operating at the full speed of the device and it is designed to perform control law computations with minimal latency. This effectively doubles the raw computing capabilities of the device. The TMU provides hardware support for common trigonometric math functions, while the FINTDIV enables fast integer division operations. The VCU adds hardware support for communications, complex math, and CRC calculations. This paper provides an overview of each of these math enhancements.

2 Floating-Point Unit (FPU)

Many control system designs typically start with simulation tools, where the algorithms are developed with floating-point math. These algorithms can then easily be ported to a microcontroller that has native floating-point math support. Floating-point math provides a large dynamic range, thereby making it easier to develop code compared to fixed-point math. The programmer no longer needs to worry about scaling and saturation. Additionally, robustness is improved since floating-point values do not wrap around the number line on an overflow or underflow, as they would in fixed-point math. These characteristics enable the high performance mathematical capabilities that are needed for advanced control systems. Also, the C2000 MCU architecture has been optimized to support high-level language programming, along with seamless support from a complete set of TI development tools.

The C2000 MCUs feature a C28x CPU that is designed around a 32-bit fixed-point accumulator-based architecture. It utilizes the best features of digital signal processors and microcontroller architectures. The addition of the FPU to the C28x fixed-point CPU enables the C2000 MCUs to support hardware IEEE-754 single-precision floating-point format operations. Devices with the C28x+FPU add an extended set of floating-point registers and instructions to the standard C28x architecture. These additional registers are: eight floating-point result registers, a floating-point status register, and a repeat block register. The repeat block adds zero overhead looping, which enables flexibility to the processor over the repeat single instruction. All of the registers are shadowed, except the repeat block register. Shadowing is useful with high priority interrupts for fast context save and restore of the floating-point registers.

Some C2000 MCUs are available with a FPU64 that provides hardware support for both IEEE-754 single-precision and double-precision floating-point operations. Devices with the C28x+FPU64 utilize the same registers as the FPU except for the addition of eight floating-point results extension registers for the double-precision floating-point operations. The FPU64 enhancements support all existing FPU single-precision floating-point instructions in addition to the 64-bit double-precision floating-point instructions.

The compiler tools provide C programming support for the CPU which makes it easy to write software, in addition to porting existing code. Since the FPU instructions are extensions of the standard C28x instruction set, most instructions operate in one or two pipeline cycles and some can be done in parallel. The FPU64 64-bit instructions operate in one to three pipeline cycles and some can be done in parallel, too. Floating-point performance dramatically enhances the mathematical computation horsepower used in signal processing and control algorithms.

Table 1. FPU Performance Improvements

Function	Type	FPU Cycles	FPU64 Cycles	Fixed Cycles	Improvements/Comments
Complex FFT	512 pt	24243	43935	63192	2.61x (FPU) / 1.44x (FPU64) vs Fixed Point
	1024 pt	53219	98683	141037	2.65x (FPU) / 1.43x (FPU64) vs Fixed-Point
Real FFT	512 pt	13670	20219	34513	2.52x (FPU) / 1.71x (FPU64) vs Fixed-Point
	1024 pt	30352	45476	76262	2.51x (FPU) / 1.68x (FPU64) vs Fixed-Point
Square Root	Compiler intrinsic	22	22	64	2.91x (FPU/FPU64) vs Fixed-Point – both modes use 32-bit float-point arguments
Finite impulse response (FIR)	64 pts	119	280	111	0.93x (FPU) / 0.40x (FPU64) vs Fixed-Point – FIR algorithms using circular addressing mode

3 Control Law Accelerator (CLA)

Enabling extremely high performance computation and efficient processing is critical for solving today’s complex real-time control applications. Real-time control systems require minimal latency where the time delay between sampling, processing, and outputting must fit within a tight time window in order to meet performance objectives. For example, a typical digital power controller consists of an ADC to read the input signals (e.g. voltage and current), a math engine to compute the control law algorithms (e.g. PID, 2-pole/2-zero, and 3-pole/3-zero compensators), and a PWM channel to output the calculated waveform. Many advanced control systems would greatly benefit from an architecture that integrates these functions in such a way as to minimize latency, yielding the absolute minimum sample to output delay. Ideally, this architecture would execute time-critical control loops concurrently with the main CPU and free it up to perform other required tasks. In addition, the architecture must have a built-in protection mechanism to guard against over-current and over-voltage conditions. To address these important requirements, TI developed the CLA.

The CLA is a fully-programmable independent 32-bit floating-point hardware accelerator that is designed for math intensive computations. This accelerator can offer a significant boost to the performance of typical math functions that are commonly found in control algorithms. The CLA is designed to execute real-time control algorithms in parallel with the C28x CPU, effectively doubling the computational performance. This makes the CLA perfect for managing low-level control loops with higher cycle performance improvements over the C28x CPU. Another advantage of the CLA is that since it directly accesses memory, the overhead penalty for managing a data page pointer is removed. Additionally, the multiplier on the CLA does not require any delay slots, thus providing true single-cycle performance. A device using the CLA can achieve about a 1.3 times performance improvement over the C28x CPU for applications like motor control and solar, as shown in the table below. Furthermore, by using the CLA to service time-critical functions, the C28x CPU is freed up for other tasks, such as communications and diagnostics.

Table 2. CLA Performance Improvements

	Number of Execution Cycles		
	CPU	CLA	
Application	Min/Max	Min/Max	Improvement
Motor AC Induction	888/952	639/694	1.39x (vs CPU)
Power CNTL 2p2z	48	39	1.23x (vs CPU)
Power CNTL 3p3z	68	52	1.31x (vs CPU)

The CLA is able to minimize latency because it has direct access to the various control peripherals such as the ADC and PWM modules. Utilizing this low-latency architecture and capability to directly access the various control peripherals provides a fast trigger response. The CLA is able to read the ADC result register on the same cycle that the ADC sample conversion is completed. This “just-in-time” reading of the ADC reduces the sample to output delay and enables faster system response for higher frequency control loops.

Programming the CLA consists of initialization code and tasks. A task is similar to an interrupt service routine, and once started it runs to completion. Each task is capable of being triggered by a variety of peripherals without CPU intervention. This makes the CLA very efficient since it does not use interrupts for hardware synchronization, nor must the CLA do any context switching. Compared with the traditional interrupt-based scheme, the CLA approach eliminates jitter, and furthermore the execution time becomes deterministic. It supports eight independent tasks, each of which is mapped back to an event trigger, such as a timer or the availability of an ADC result. Separate tasks can be used to support multiple control loops or phases at the same time.

Some C2000 devices feature an enhanced version of the CLA with the option of running the lowest priority task as a background task. Once triggered, it runs continuously until it is terminated or reset by the CLA or MCU. The remaining tasks in priority order can interrupt the background task when they are triggered. If needed, portions of the background task can be made uninterruptible. Typical uses of the background task include running continuous functions, such as communications and clean-up routines.

Another key benefit of the CLA, over hardware-based control law implementations, is flexibility. The CLA is a fully software programmable solution where developers can freely modify their control system without the time and high cost required to redesign a hardware-based solution. Also, the CLA is significantly more power-efficient in executing operations when compared to the main C28x CPU which is an advantage for power-sensitive applications.

4 Trigonometric Math Unit (TMU)

The TMU is an extension of the FPU and enhances the instruction set of the C28x+FPU by efficiently executing trigonometric and arithmetic operations that are commonly used in control system applications. Similar to the FPU, the TMU is an IEEE-754 floating-point math unit tightly coupled with the CPU. However, where the FPU provides general-purpose floating-point math support, the TMU focuses on accelerating several specific trigonometric math operations that would otherwise be quite cycle intensive. These operations include sine, cosine, arctangent, divide, and square root. Some C2000 devices include an enhanced version of the TMU for supporting nonlinear PID applications. Additional instructions have been added for efficient computation of logarithm and inverse exponent operations which are used in the nonlinear control law. The TMU instructions include:

Table 3. TMU Supported Instructions Summary

Operation	C Equivalent Operation
Multiply by 2*pi	a = b * 2pi
Divide by 2*pi	a = b / 2pi
Divide	a = b / c
Square Root	a = sqrt(b)
Sin Per Unit	a = sin(b*2pi)
Cos Per Unit	a = cos(b*2pi)
Arc Tangent Per Unit	a = atan(b)/2pi
Arc Tangent 2 and Quadrant Operation	Operation to assist in calculating ATANPU2
Logarithm	a = $LOG_2(b)$
Inverse Exponent	a = $2^{-\|b\|}$

The TMU uses the same pipeline, memory bus architecture, and FPU registers as the C28x+FPU, thereby removing any special requirements for interrupt context save or restore.

The C2000 compiler has built-in support that allows automatic generation of the TMU instructions. The user writes code in C using math.h functions, and the compiler uses the TMU instructions, where applicable, instead of run-time support library calls. This results in significantly fewer cycles and dramatically increases the performance of trigonometric operations.

The TMU can have a significant impact on many commonly used real-time control algorithms such as:

- Park and Inverse Park Transforms
- Space Vector Generation
- dq0 and Inverse dq0 Transforms
- FFT Magnitude and Phase Calculations

For example, a Park Transform typically takes anywhere from 80 to more than 100 cycles to execute on the FPU. With the TMU a Park Transform takes only 13 cycles, yielding an 85 percent improvement as compared to without the TMU.

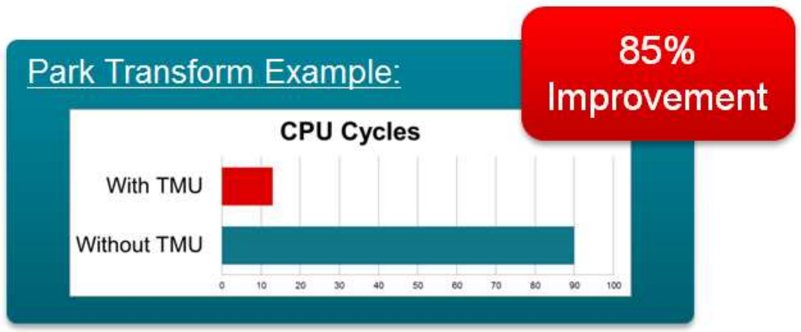

Figure 2. TMU Performance Improvement for Park Transform Example

In a typical system application, such as digital motor control (AC induction and permanent magnet) and 3-phase solar applications, about a 1.4 times performance improvement can be achieved using the TMU over just the FPU.

Table 4. TMU Performance Improvements

Application	Number of Execution Cycles		Improvement
	FPU	TMU	
	Min/Max	Min/Max	
Motor AC Induction	888/952	593/670	1.42x (vs FPU)
Motor Permanent Magnet	783/786	547/592	1.32x (vs FPU)
Solar 3-Phase	1351/1358	985/983	1.38x (vs FPU)

An existing C28x design can realize an immediate advantage using the TMU without the need to rewrite any code. Simulation-based generated code can realize the same benefits. Portability is maintained since the same code can be used on TI MCUs with and without the TMU support.

5 Fast Integer Division Unit (FINTDIV)

The FINTDIV extended instruction set optimally supports fast division operations commonly found in adaptive control systems for scaling parameters based on a variable. All instructions execute in a single cycle and three types of integer division are supported (Truncated, Modulus, Euclidean) of varying data type sizes (16/16, 32/16, 32/32, 64/32, 64/64) in unsigned or signed formats. Truncated format is the traditional division performed in C language (where “/” is the integer, and “%” is the remainder); however, the integer value is non-linear around zero. Modulus and Euclidean formats are more appropriate for precise control applications because the integer value is linear around the zero point, and this avoids potential calculation hysteresis. Both the Modulus and Euclidean divisions are supported by C intrinsics, and the C28x compiler supports all three division formats for all data types. Since the FINTDIV uses the existing FPU register set to carry out the FINTDIV operations, there are no special considerations relating to interrupt context save and restore.

Table 5. FINTDIV Performance Improvements

Operation	Number of Execution Cycles		Improvement (vs CPU)
	CPU (‘/’ C operator)	FINTDIV (intrinsics)	
i16/i16 Truncated	52	16	3.3x
i16/i16 Euclidean and Modulus	56	14	4.0x
u16/u16	56	14	4.0x
i32/i32 Truncated	59	13	4.5x
i32/i32 Euclidean and Modulus	63	14	4.5x
i32/u32 Truncated	37	14	2.6x
i32/u32 Modulus	41	14	2.9x
u32/u32	37	12	3.1x
i32/i16 Truncated	60	18	3.3x
i32/i16 Euclidean and Modulus	64	16	4.0x
u32/u16	38	13	2.9x
i64/i64 Truncated (1)	78 – 2631	42	1.9x – 62.6x
i64/i64 Euclidean & Modulus (1)	82 – 2635	42	2.0x – 62.7x
i64/u64 Truncated (1)	54 – 2605	42	1.3x – 62.0x
i64/u64 Euclidean & Modulus (1)	58 – 2609	42	1.4x – 62.1x
u64/u64 (1)	53 – 2548	42	1.3x – 60.7x

(1) FINTDIV implements 64-bit integer division that is optimized in a fixed number of cycles for deterministic behavior. Without the FINTDIV acceleration enabled, 64-bit integer division is implemented with generic CPU instructions and the number of cycles can vary significantly based on the value of the numerator and denominator.

6 Viterbi, Complex Math, and CRC Unit (VCU)

Todays advanced control systems, such as motor control and power applications, can benefit from intelligent management and communications to optimize efficient operation. Power line communications (PLC) has become an ideal solution for intelligent management since the existing infrastructure can be used cost effectively. Communicating data in noisy environments is very challenging and computationally intensive. A typical microcontroller running a control application at its limit cannot tolerate the additional burden of supporting power line communications, and may require an additional processor. To solve this problem, TI developed the VCU. The VCU is a tighly coupled fixed-point unit that improves performance of communications-based applications by a factor of roughly seven times. Additionally, cost savings are realized by eliminating the need for a separate processor. Besides communications, the VCU is very useful for general-purpose signal processing applications such as filtering and spectral analysis. For example, spectral analysis can be used to process motor vibration noise to determine the impact of vibration on a system, estimate the motor operating life, and calibrate the control loop to improve efficiency.

The VCU has been designed to be flexible in supporting various communications technologies. For the typical MCU, four key operations consume most of the processing power: Viterbi decoding, complex Fast Fourier Transform (FFT), complex filters, and Cyclical Redundancy Check (CRC). Using the hardware capabilities of the VCU, an application will significantly benefit by the increased performance over a software implementation. As an example, the performance contributions of each key operation are:

- Viterbi decoding is commonly used in baseband communications applications. The Viterbi decode algorithm consists of three main parts – branch metric calculation, add-compare-select (Viterbi butterfly), and traceback operation. With the VCU, the branch metric calculation can be completed in a single cycle (code rate = 1/2, and two cycles for code rate = 1/3). The Viterbi butterfly takes 2 cycles per stage, as compared to 15 cycles per stage without the VCU. The traceback takes 3 cycles per stage, as compared to 22 cycles per stage without the VCU.
- The complex FFT is used in spread spectrum communications, as well as many other signal processing algorithms. For a 16-bit fixed-point complex FFT the VCU only requires 5 cycles per stage, as compared to approximately 20 cycles per stage without the VCU.
- Complex filters are used to improve data reliability, transmission distance, and power efficiency, and are commonly used in other various signal processing applications. The VCU can perform a complex I and Q multiply with coefficients (four multiplies) in a single cycle, as compared to approximately 10 cycles without the VCU. In addition, the VCU can read/write the real and imaginary parts of 16-bit complex data to memory in a single cycle.
- CRC algorithms are used for verifying data integrity over large data blocks, communication packets, or code sections. The VCU can perform 8-bit, 16-bit, 24-bit, and 32-bit CRCs completely in the background, offloading the main C28x CPU. For example, the VCU can compute the CRC for a block length of 10 bytes in 10 cycles, as compared to approximately 250 cycles without the VCU. A CRC result register contains the current CRC and is updated each time a CRC instruction is executed. This simplifies the CRC calculations and access to the final CRC value.

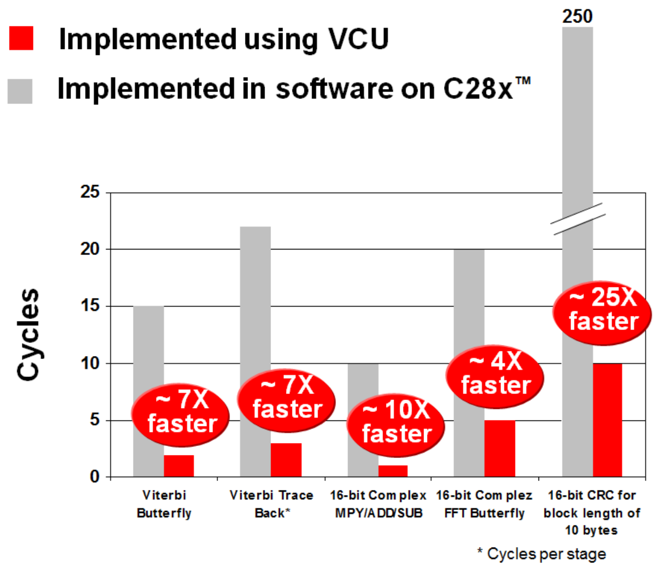

Figure 3. VCU Performance Improvements Compared to Software-Only Implementations

Devices with the C28x+VCU add an extended set of registers and instructions to the standard C28x architecture, which are used to support the acceleration of communications-based algorithms. The additional registers are: nine result registers, two traceback registers, a configuration and status register, and a CRC result register. The VCU performs fixed-point operations using the same existing instruction set format, pipeline, and memory bus architecture as C28x.

Programming the VCU is made easy with TI's C2000Ware software suite. TI provides a complete library of C-callable assembly functions. These functions are implemented using the VCU instruction set to optimize efficiency and minimize overhead. TI also provides higher-level functions to support PLC communications standards such as PRIME and G3.

Some devices utilize a dedicated cyclic redundancy check unit (VCRC) rather than the full featured VCU for applications not requiring Viterbi decoding or complex math support. This enhanced VCRC is an extension of the C28x CPU and it includes registers and instructions to support CRC algorithms. CRC algorithms provide a straightforward method for verifying data integrity over large data blocks, communication packets, or code sections. The VCRC can perform 8-bit, 16-bit, 24-bit, and 32-bit CRCs, and it is capable of computing the polynomial code checksum for a block length of 10 bytes in 10 cycles (a byte of data in a single cycle). For custom CRC polynomials the execution time increases to three cycles. A CRC result register contains the current CRC, which is updated whenever a CRC instruction is executed.

7 Summary

Utilizing the high performance C28x CPU along with the advanced hardware math enhancements described in this paper, the TI C2000 family of MCUs provides the advanced processing power required for today’s complex real-time control systems. Combining these enhancements with the various control-optimized peripherals, such as high-speed ADCs and high-resolution PWMs, engineers can minimize latency while increasing system performance. TI provides a comprehensive set of development tools and software that enable engineers to quickly design, test, and produce extremely reliable control systems. A wide range of TI C2000 MCUs are available to solve the most demanding control system requirements.

The C2000 MCU family includes a wide array of devices that have been designed for both high performance and low-cost real-time control applications. Based on an extremely fast C28x CPU, advanced control peripherals, and integrated analog functions, the C2000 MCUs can reduce system cost while increasing system reliability. Combining the CPU with the CLA running concurrently can effectively double the throughput of the device. Additionally, some family members feature a dual-core microcontroller, and when combining each CPU with its own CLA, the device has the capability for delivering the equivalent of up to four times the performance of a single CPU. Conversely, other family members feature a high level integration of control and analog peripherals for reducing system complexity and offers greater efficiency for cost-sensitive designs.

The C2000 family of MCUs is ideal for applications requiring advanced real-time signal processing such as industrial drives, digital power, renewable energy, smart sensing, white goods appliances, motor control, electric vehicle and hybrid electric vehicle (EV/HEV).

8 References

For additional information about the C2000 MCU family, see the TI web site at:

- http://www.ti.com/c2000

The availability of the various math units and peripherals on each device can be found in the following document:

- Texas Instruments: *C2000 Real-Time Control Peripheral Reference Guide*

For detailed information about the CLA, see the device-specific Technical Reference Manual.

The extended instruction sets for the FPU, TMU, FINTDIV, VCRC, and VCU can be found in the following document:

- Texas Instruments: *TMS320C28x Extended Instruction Sets Technical Reference Manual*

Details about the FPU, TMU, and FINTDIV intrinsics for providing ease of software development can be found in the following document:

- Texas Instruments: *TMS320C28x Optimizing C/C++ Compiler v20.2.0.LTS User's Guide*

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from A Revision (November 2016) to B Revision	Page
• The title of this document was updated.	1
• Update was made in the Abstract of this document.	1
• Updates were made in Section 1.	2
• Updates were made in Section 2	2
• Updates were made in Section 3.	3
• Updates were made in Section 4.	5
• Added new Section 5.	6
• Updates were made in Section 6.	7
• Updates were made in Section 7.	9
• Added new Section 8.	9