TMS320C27x

New Generation Of Embedded Processor Looks Like a µC, Runs Like a DSP

WHITE PAPER: SPRA446

Alex Tessarolo
TMS320C27x Chief Architect

Digital Signal Processing Solutions
March 1998

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI and Real-Time Data Exchange are trademarks of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

New Generation Of Embedded Processor Looks Like a µC, Runs Like a DSP

Abstract

The Texas Instruments (TI™) TMS320C27x (C27x) digital signal processor (DSP) is a new kind of DSP that combines the math-intensive tasks only a DSP can perform efficiently with the ease of use and code efficiency of a general-purpose processor. The C27x is the industry's most code-efficient engine and enables users to integrate the functions of both types of processors into a single architecture.

The C27x is designed for embedded systems containing DSPs and general-purpose processors like microcontrollers. Examples are computer peripherals such as high-performance hard drives, DVD players, and other mass storage devices, industrial systems like robotics and electronic meters, and consumer electronics, such as Internet phones. In systems such as these, integrating two processors into one can reduce both the size and cost of a product and help meet the considerable demand for increased performance, enhanced features, and shorter development time.

A major challenge for designers of high performance embedded systems is debugging today's highly integrated devices. As speeds increase and more functions are built into a single piece of silicon, it becomes difficult to have visibility into processor operation. For this reason, the C27x is the first architecture to feature the next generation emulation technology; built-in real-time debug and high-speed Real-Time Data Exchange.

Product Support

World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Email

For technical issues or clarification on switching products, please send a detailed email to dsph@ti.com. Questions receive prompt attention and are usually answered within one business day.

Introduction

Many embedded systems existing in the industry today contain two processors performing very distinct tasks. One processor, such as a DSP, performs the algorithm intensive task and the other processor performs the more general-purpose tasks, such as command handling, user interface, diagnostics etc.

High-performance hard disk drives, DVD players, and other mass storage devices are examples of such systems. Many similar systems are out there in the industry; however, for this discussion, we refer to the disk drive as an example.

Three on-going trends in the electronics industry dictate if a product or company will survive in a very competitive world. Continual cost reduction of system electronics, an ever-present hunger for more features to differentiate a product, and ever shrinking product life cycles.

These trends force designers to integrate more and more electronic functions onto a single chip, hence creating more "systems on a chip". They drive the increase in processing performance, hence higher MIPS. They drive increased software demands to enhance features or replace expensive hardware with cheaper alternatives. And they force designers to get new or enhanced products to market faster, shortening development time.

The above trends force users to re-examine their dual-processor systems and look for ways to reduce cost. Two of the more attractive options are:

- Replacing two processors with one high performance processor
- Integrating two processors into a single piece of silicon

Replacing two processors with one higher performance processor requires that the processor be able to perform the math intensive algorithmic tasks and the general processing tasks equally well. The processor must have the processing power of a DSP, yet the code efficiency of a micro-controller or other general processing engine.

In some systems, replacing two processors operating in parallel with a single higher performance processor may not be the ideal solution and may compromise system response time. In this case, the optimal solution is to replace two different processors with two identical processors. This greatly simplifies ASIC design, code development, and debug since both processors are identical.

Hence, for both solutions, we require a processor that has the power of a DSP but the ease of use and code efficiency of a general-purpose processor.

Did such a processor exist? The answer is yes and no. DSP devices, such as the TI TMS320C2xx (C2xx) core, have been used as both a DSP engine and a general-purpose processing engine but at the penalty of code size. Code efficient RISC processors are available but do not have DSP performance levels, which limits the scope of the device. Other devices have attempted to add DSP-like performance by adding multiply and accumulate (MAC) functions to an existing core. These are Band-Aid solutions that cannot fully match the performance of a true DSP.

What was needed was a processor representing the best of both worlds: the processing performance of a DSP with the ease of use and code efficiency of a general-purpose engine. That was the genesis of the TMS320C27x architecture.

The C27x DSP is a new kind of DSP that not only performs the math-intensive tasks only a DSP can perform efficiently but is also easy to use and has the industry's most code-efficient engine. Users can integrate one or more such processor cores into a single piece of silicon and perform all of the necessary tasks with equal efficiency.

MCU versus DSP

What makes a DSP and what makes a general-purpose processor, or MCU? The boundaries between the two can sometimes be blurred. All processors can perform digital signal processing functions. It is all a question of signal bandwidth. A true DSP processes much higher signal bandwidths than is otherwise possible with traditional MCU architectures.

In fact, the difference can represent several orders of magnitude. It is probably easier to look for the specific features that make a processor a true DSP. A processor that lacks these features can be lumped in the MCU basket. The following features are those that make a DSP:

- Harvard bus architecture

  DSP algorithms are very data intensive; hence, the ability to feed up to two data streams to the processing unit is essential in many DSP algorithms (such as filters). Separating the program and data buses also removes the instruction-fetching bottleneck. Hence, a Harvard bus architecture, which separates the program and data buses, allows instructions and data to be processed concurrently for improved throughput and hence performance. Compared to the single bus architecture of most MCUs, a DSP typically has two read buses and one write bus. The two read buses can fetch an instruction and a data value or two data values in a single operation and at the same time perform a write operation. The performance of a DSP can be up to three times that of any MCU with only a single bus.

- Pipelined operations

  To make use of the multiple buses and achieve the greater performance levels, all DSPs implement a multi-stage pipeline, in many ways similar to RISC MCUs. Pipelining enables the more efficient use of on-chip silicon resources, allows multiple operations to occur, and enables much faster cycle times than possible in a CISC-type architecture.

- True single-cycle multiply and accumulate

  Some MCU devices include MAC units. However, many of these units may perform a multiply and accumulate in a single cycle only once a value is loaded into a register. The time it takes to load two data operands into a register must be taken into account. A true DSP is able to multiply two data values directly from memory and accumulate the result in a single cycle.

- Special addressing modes, such as circular addressing.

  Many DSP algorithms require the processed data array to behave like a delay line. A method of achieving delay lines is to implement circular addressing. In circular addressing, a pointer into memory automatically wraps around to the beginning of a data array when the specified end of the array is reached. Such operations are very common in filters (which are a very common DSP algorithm). To implement such operations on a device without a circular addressing mode requires significant cycle and code overhead.

- Special instructions and modes such as saturation operations.

  In many control-based algorithms, the designer must make sure that calculations fit within real limits. If, for example, the output generated by a calculation is to be fed to a 10-bit D/A (digital-to-analog) converter, the calculated output must be saturated to the numerical limits of the D/A converter. This is termed saturation and is a common operation in many control algorithms. A DSP has specific register modes and instructions that perform this operation automatically or within a few cycles. To implement this otherwise requires many instructions.

The above list does not exhaust all of the unique features that a true DSP device can offer but are some of the common operations that a designer may use in control-based applications (for example, servo control of a disk drive). All of the above add up to a significant performance advantage over an MCU minus the above features. A DSP can offer an order of magnitude better performance than any MCU.

Some MCU examples of C27x competitors are

- Intel 80C196, 80186, 80286
- Motorola HC11, HC16, Cold-Fire, M-Core,
- Advanced RISC Machines, ARM7, ARM7TDMI, ARM9, ARM-Piccolo
- Hitachi SH, SH3, SH-DSP
- Siemens C166, ST10/20, Tri-Core

But if DSPs are such great performers, why don't they rule the world? Traditionally, DSPs were optimized to perform number crunching very well because that is what designers wanted from a DSP. DSPs typically lacked the following compared to an MCU:

- Ease of programming

  The DSP instruction set can sometimes look cryptic to traditional MCU users. Pipelining can be alien to some users, for example, hardware stacks instead of software stacks).

- They were designed to do math operations very well, but logic and I/O operations, although supported, usually required more instructions than traditional MCUs. Atomic operations were lacking.

- Limited address reach

  Because some DSPs operated on very small programs, address reach was typically limited to 64K.

- Code efficiency was compromised for performance. Since DSP algorithms were not large, code efficiency was not a major care about.

- Interrupt latency and context switch performance was not great. Since most DSP tasks are synchronous to interrupt events and perform batch operations, this was not a factor in their designs.

There is no reason why a DSP cannot perform the above tasks efficiently or that such operations cannot be supported by a DSP architecture. They just weren't the focus until now. The C27x architecture addresses all of the above areas.

Why is code efficiency important? As processor speeds increase, more and more of the processor memory resources will be integrated on-chip. It becomes too expensive and performance limiting to execute from external memory. As more and more memory is integrated on-chip, the cost of the chip starts to be dominated by the amount of memory integrated. An inefficient engine means more memory to integrate or less program able to run at full speed within the chip.

There are other side benefits to improved code efficiency: one is improved performance due to reduced instruction fetches and cycle counts. Another is improved power consumption due to reduced memory accesses. Hence, code efficiency is becoming a keyconsideration.

TMS320C27x Goals

The TMS320C27x core architecture was designed with two primary goals: overcome the DSP's weaknesses in embedded applications and address the targeted application requirements to satisfy customer needs. These goals are outlined as follows:

- Goals to address traditional DSP weaknesses
  - Improved code efficiency for Assembly and high level languages, such as C
  - Improved interrupt and context switch response
  - Increased linear address reach breaking past the 64K barrier
- Goals to address targeted application needs
  - Efficient compiler with minimal overhead over Assembly code
  - 100+MIPS DSP class performance
  - Real-time emulation capabilities

All of the above goals had to be met while keeping the core size as small as possible. How small? We set a target core size of $3mm^2$, which is competitive with existing cores. Additionally, core size is becoming less dominant in overall chip cost because of the large memory blocks being integrated. It is more important to minimize the size of the memory blocks by improving code efficiency, than to worry about the total size of the core.

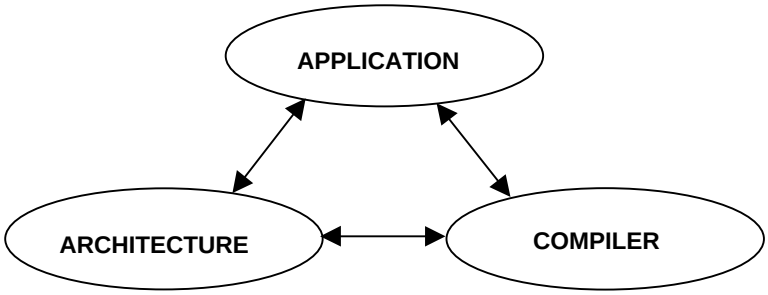
The goals of improved code efficiency and efficient compiler engine sound the same but basically highlight the issue that the architecture and compiler must complement each other to produce the best possible results. You can have a good compiler but poor architecture or vice versa. Additionally, if we take the view that the compiler is a "dumb" assembly language programmer, by improving C code efficiency we naturally improve assembly code efficiency.

The development of the C27x architecture was heavily driven by analyzing C code and making architectural tradeoffs based on the impact to compiler efficiency. This was possible because of two factors:

- We had an internal compiler technology for which we could spin a new compiler for a prototype architecture within a very short time. This gave us almost instant feedback on what works and what does not.

- We had a large collection of real user code from Mass Storage and other applications that we could use to benchmark the architectural tradeoffs based on the prototype compilers.

Essentially, to design the best architecture for a targeted application you must close the loop on the following triangle:



Code Efficiency

Traditionally, DSPs have been designed to optimize performance. The instruction set and associated tools, such as compilers, were always tuned to generate fast code, usually to the detriment of code size. On the C27x, the focus is toward code efficiency without compromising performance.

As a start, one must look at what a good compiler likes from the architecture point of view. Compilers are traditionally designed for load/store architectures where all operations are performed on registers and the compiler manages a fixed set of resources. Compilers like to have many registers because it gives the compiler multiple resources to work with and hence minimizes the painful spilling of registers. Compilers like orthogonality. They like registers to operate identically so that the compiler is free to allocate register resources without worrying about restrictions.

Most RISC architectures include many orthogonal registers or, in some cases, two sets of orthogonal registers: one set for addressing and one set for data processing. Traditionally, compilers designed for RISC-type machines (such as the ARM7TDMI) produce very good code efficiency.

On the other hand, DSP-style problems are not heavy register users. Typically, a DSP algorithm, such as a filter, sources its data directly from memory and accumulates the resultant operation in one or two registers. The result is then stored back to memory. This type of operation is not comprehended well by the traditional Load/Store view of the Compiler; hence, an architecture that supports operations directly from memory requires the compiler to be specially tuned to perform such operations efficiently. For a Load/Store architecture to perform such DSP operations efficiently, it must support Load/Store operations in parallel with DSP operations (such as MAC).

For the C27x architecture, the above represented a problem because we needed to retain the DSP look and feel of the earlier generation architecture, the C2xx core, for compatibility reasons. This architecture was accumulator-based whereby a single 32-bit register performed most of the work. This single register was a serious bottleneck to the compiler. We addressed the problem in the following ways:

- After examining much of the application code, we found that 16-bit data types were the most commonly used. Hence, we split the accumulator register into two independent 16-bit registers. This had a significant impact on compiler efficiency.
- The C2xx architecture had no register-to-register operations. Register-to-register operations are included in the C27x architecture.
- For DSP-style problems, the compiler engine was tuned to look for DSP-style code patterns and optimize the code according to the capabilities of the architecture. We could do this because of the ability built into the TI compiler technology.

Nevertheless, the above did not improve code efficiency enough. Additional observations were made:

- When examining many embedded applications, a frequent operation to peripherals or variables was as follows:

```
load      reg,mem       ; read
operate   reg,operand   ; modify
store     reg,mem       ; write
```

  These operations could account for approximately 20% of the code. In most cases, the value stored in the register was not re-used. Therefore, if we could implement a single instruction operation that performed the read-modify-write operation directly on a memory location without using a register resource, we would obtain significant code savings.

- 10% of the application code was conditional jumps. Almost 90% of conditional jumps were within a distance of ±128 instructions. On the C2xx architecture, we used a 32-bit instruction for conditional jumps. Reducing this to a single 16-bit word operation would gain significant code savings.
- Compilers use the stack for passing parameters and for storing temporary variables. Our analysis showed that 95% of function calls used 16 words or less of stack area per call frame. The ability of the C2xx to manage the stack was not strong and we needed to improve this and/or add more flexible registers to allow for register passing models.

We addressed the above observations in the following ways on the C27x architecture:

- We added a new class of instruction operations called Read-Modify-Write, which can perform an

  ADD/SUB/INC/DEC/AND/OR/XOR

  operation directly on a 16-bit memory location without needing a register. This effectively makes every memory location look like a register, making the architecture look like a "virtual" register machine.
- We added a 16-bit short conditional jump instruction with a ±127 word reach.
- We added a new addressing mode called stack indexing with a fixed offset that can read or write to any location in the stack within a 64 word range from the top of the stack. This dramatically improves the ability of the compiler to access parameters and allocate temporary values. To improve the register passing model, we added register-to-register operations. The compiler now can efficiently work with stack and register values.

The above architectural enhancements, along with 90% of the instructions being 16-bit opcodes and the improved addressing modes, helped us achieve the following breakdown of code savings for the C2xx architecture:

Register-to-Register Operations + Multi 16-Bit Registers	10%
Read-Modify-Write Operations	15%
Short Conditional Jump Instruction	10%
Stack Indexing + Improvements in Addressing Modes	10%

Total Improvement over C2xx Architecture	45%

Given that we had benchmarked the C2xx architecture versus many other devices and found that it typically generated 20-30% more code than other devices, we knew that the C27x architecture would offer us anywhere from 15-25% advantage over competitive devices. Subsequent benchmarking showed us a 15-20% code size advantage over the ARM7TDMI, the class leader, for Mass Storage application code benchmarks.

Performance

DSPs, with their Harvard bus architecture and instruction pipelining, are inherently high performance machines. This allows a DSP to fetch instructions and perform data read and/or data write operations in parallel. For that reason, this basic form of the architecture is retained on the C27x. The primary issue we faced in designing the C27x architecture was how to achieve 100MHz plus operation in today designs when the primary bottleneck in increasing processor performance is memory speeds. We can design cores that execute much faster than the large memories we need to fetch instructions or perform data read or write operations from. Four architectural design alternatives address this issue:

- Caching

  There are many possible varieties of cache mechanisms. Their effectiveness is based on the targeted application. For some applications a cache has no effect on performance. Hence, it is wiser to design a core whereby a cache is implemented as an external function to the core and the cache specifics can be tuned to the application needs. Hence, caching is supported as an external function to the C27x core.

- Pipelined Memory Accesses

  Given that we have applications where caching is not effective, the core must directly connect to large memory banks. To improve memory access speeds, we need to pipeline memory accesses. On the C27x we implemented two stage pipelining on all memory accesses. The address is thrown out on one cycle and the data is read or written on the next cycle. This enables the C27x to achieve clock speeds much higher than shallower pipeline designs. The penalty is a deeper pipeline, which penalizes discontinuities. This is compensated for by the greater code efficiency, which reduces instruction fetches, and by moving some common operations to earlier stages in the pipe. In the balance, achieving higher clock speeds more than compensates for the hit to discontinuities.

- Wider word fetches

  Because in many applications large amounts of memory are integrated on-chip, it is possible to organize memories to have large word widths. Larger word widths allow multiple instructions or data to be read in a single operation. Although this may take more cycles, it can be operated on at full speed once the data is within the CPU. However, the penalty for wider word fetches is the cycle penalty that results if a discontinuity is not aligned to the beginning of the word.

This can be avoided by alignment, which penalizes code efficiency. Therefore, you can make a memory only so wide before it starts to penalize either performance or code efficiency. On the C27x all fetches are 32 bits wide. Because 90% of the instructions are 16 bits, every fetch would contain two instructions (in most cases). This enables the C27x to execute from 20ns memory and achieve near 100MIPS performance or, alternatively, it allows 32-bit instructions to execute single cycle from 0-wait memory. Also, the data read and write buses are 32-bits wide. This allows for single-cycle, 32-bit read/write operations to memory.

- Decoupled Fetch Stage With Pre-Fetch Queue

  By decoupling instruction fetches from data read/write operations, it allows instructions to be fetched into a pre-fetch queue while data reads and writes may be stalled. This improves performance when accessing slower memories or peripherals. The C27x implements a decoupled, four-deep, 32-bit word pre-fetch queue.

One aspect of the performance options considered was the question of a protected or unprotected pipeline. An unprotected pipeline allows operations to be scheduled to overcome pipeline delays and hence improve performance. The drawback is an increased likelihood of difficult-to-find bugs, which makes it difficult for users moving from traditional general-purpose processors to use such a device. Hence, it was decided to implement a fully protected pipeline to facilitate an easier-to-use processor that is also much easier to debug. All operations on the C27x are thus scheduled as written. The core automatically handles any memory or register conflicts.

Interrupt Response and Context Switch

The one area that DSP designs typically have not supported well is interrupt context save and restore performance. Generally, most DSP tasks are batch-driven and the DSP operation is typically the primary operation. To improve interrupt context save/restore performance, the following options were considered:

- Bank of shadow registers

  This method is implemented on several traditional DSPs, such as the TMS320C5x and the ADSP-21xx from Analog Devices. Most core CPU registers are automatically copied to a set of one-deep shadow registers at the entry of every interrupt. The shadow registers are restored on exiting an interrupt. Such a method can only be effective for the highest priority interrupt.

- Multiple register banks

  This method is common among some RISC processors that have a complementary set of registers switched to on an interrupt. Again, this method is limited only to the highest priority interrupt.

- Automatic context save

  This method saves processor registers while the processor is flushing the pipeline, fetching the interrupt vector and re-building the pipeline at the vector destination. This method is attractive on a multiple bus architecture (such as the C27x) because the data read/write busses would otherwise be idle during this process.

The C27x architecture uses the automatic context save option. During the 8 cycles it takes to respond to an interrupt, almost all of the processor registers are saved. Additionally, the C27x performs interrupt management tasks, such as saving the current interrupt enable register, that otherwise must be performed by the user in the Interrupt Service Routine. Trap operations also perform an automatic context save. A special return instruction automatically restores the registers saved during the interrupt service. Therefore, the C27x can respond rapidly to interrupt requests.

Fast interrupt service helps system response and enables users to adopt new processing techniques. Time slicing of processor resources can be implemented effectively by an Operating System by minimizing task-switching time. Time slicing allocates a fixed number of time slices to individual tasks. This technique spreads the processor MIPS over multiple tasks while improving system response time.

Extended Address Reach

Traditional fixed-point 16-bit DSPs have had a limited address reach of 64K (16 address lines). To increase the appeal of a new processor and address the ever-increasing hunger for more features, we needed to increase the address reach beyond 64K.

Various paging options were considered but dismissed. Paging is inexpensive on silicon but difficult to debug and has additional code overhead to maintain paging information. Paging is not user friendly. We therefore decided to support a linear address reach of up to 22 bits for program and data space. The C27x, being a 16-bit addressable machine, supports a total address reach of 4MWords for program and 4MWords for data space.

Why 22 bits? Because this is the total number of address bits we could support while maintaining 90% of the instructions within a 16-bit opcode. Going beyond this would have required more 32-bit instructions. This would have increased the core size and cost. For the applications we target, 22-bit address reach is more than sufficient.

Some addressing modes were limited to the lowest 64K of data memory space. In many applications, most of the peripherals and critical data fit well within a 64K page. This helps minimize register size and hence silicon core size. It also helps code efficiency since the lower 64K of data memory space is supported by efficient 16-bit instructions. For program space, no such restrictions apply. Programs can be located anywhere in the 22-bit address reach without loss of efficiency.

Real-Time Emulation

One of the critical requirements in many embedded applications that control mechanical systems is the ability to debug or monitor system performance without breaking the control loop. For instance, on a disk drive, a user likes to monitor the behavior of the servo control of the actuator arm without breaking the control loop. Stopping the processor is out of the question because the actuator arm will go out of control and potentially damage the drive components.

Traditional approaches to this problem linked software monitors on the target device that communicated back to a host PC via a serial port or via the JTAG port. This technique has several limitations: it limits the data throughput, is intrusive because it requires processor resources in terms of cycles and memory, and is not robust. Also, buggy code can corrupt the monitor resources.

A new emulation technique was adopted for the C27x. This new technique uses DMA accesses directly to memory and registers to monitor system activity. The DMA looks for unused bus cycles to gain access. On a multiple bus system, such as found on the C27x, dead bus cycles are common and frequent. The emulation system can thus read and write to memory and monitor processor activity without affecting the operation of the target application code. The DMA data is imported/exported via the JTAG port. To improve data transfer speeds, the JTAG port implements data streaming.

The C27x emulation block also incorporates two breakpoints, one watchpoint, benchmarking counters, and Real-Time Data Exchange (RTDX™). RTDX allows extremely fast data transfers to/from the C27x via the JTAG port or via two special emulation pins. Data transfer rates of up to 30+MBits/sec are expected.

The C27x core also supports a visibility port that gives full visibility into the operation of the core and allows extending the on-chip debug features outside the core. This enables TI to customize emulation features based on customer and application needs. More systems on a chip and faster processor speeds are pushing designers to place more emphasis on on-chip debug features. Traditional In-Circuit-Emulation (ICE) techniques no longer work at speeds greater than 30MHz. Also, there are typically fewer pins available than the signals we wish to observe internally. Hence, high-speed serial export techniques, such as the JTAG port, are necessary. The C27x offers designers plenty of flexibility in customizing emulation to overcome the visibility limitations of today’s highly complex designs.

C27x – The Best Of Both Worlds

In conclusion, the C27x architecture not only has the number crunching capability of a DSP engine but also offers the ease of use and code efficiency of a general-purpose processing machine. The fast interrupt context save, 100MIPS (100MHz) performance increasing to 200MHz by the year 2000, and the extended address reach further enhance the capabilities of the processor. To round out the features, the C27x is equipped with the latest generation of emulation capabilities. The ability to monitor the operation of the processor, modify registers and memory contents, and single step through non-time critical code while time critical operations continue to function, offers unprecedented emulation capabilities without disturbing the application code.

The C27x is designed with the necessary features to address the problems faced by today's highly integrated embedded designs.