Infineon Carmel DSP 10XX and 20XX cores

EDN Staff - March 30, 2000

The Carmel DSP core is a family of licensable, fully synthesizable, 16-bit, fixed-point DSP cores. The first two members of the family are the 10XX and 20XX cores. Both cores operate from 1.2 to 2.7V and have low-power features. The Carmel DSP configurable-long-instruction-word (CLIW) architecture delivers very-long-instruction-word (VLIW) performance without sacrificing power-dissipation and code-compactness requirements. You can customize instructions through CLIW and provide a high degree of parallelism with the ability to simultaneously generate four addresses and perform four arithmetic operations and two data transfers. However, the compiler provides no support for this function, and you must hand-code CLIW instructions into your program. The Carmel DSP 20XX is binary-compatible with the 10XX and allows you to further modify and extend the core's instruction set.

Carmel DSP's modified Harvard architecture has separate program- and data-memory space. Carmel DSP features four 16-bit data buses and a 48-bit bus for reading four operands and two 24-bit instructions in a single cycle. The 10XX has six computation units that operate in parallel. These units include two 16×16-bit multiply-accumulate (MAC) units, two 40-bit ALUs, a 40-bit barrel shifter, and a 40-bit exponent unit supported by six 40-bit accumulators. The 20XX can have as many as 10 computation units that operate in parallel. These units can include four 16×16-bit MACs, two to four 40-bit ALUs, a 40-bit barrel shifter, a 40-bit exponent unit, and other accelerators for applications such as a 32-bit MAC for audio processing or quad 8-bit MACs for video processing.

A programmer may split each accumulator into two 16-bit half-accumulators, which the program can use as a source or a destination. In addition to accumulators, a program can fetch operands directly from memory and route them to either of the execution units. The program can also write the results directly to the same memory location without requiring a temporary register (that is, read-modify-write). This non-load-store design avoids the overhead typical of load-store architectures.

The ALUs operate on 40-bit data and provide arithmetic and logic operations back-trace support, which can accelerate Viterbi algorithms and saturation and limit support. The MAC units operate on all combinations of signed/unsigned, 16-bit operands. Also, the MAC units can perform addition and subtraction operations, which allow the Carmel to complete four additions or subtractions per clock cycle.

The barrel shifter supports both arithmetic and logical shifts by 0 to 40 bits left or right and rotate right through carrying 16-, 32-, or 40-bit operands. The exponent unit handles exponent detection and block floating point.

Addressing modes—The Carmel DSP family supports direct, indirect, index-by-register, index-by-immediate, and short and long addressing modes. Both cores support linear, bit-reversal, modulo, and special-modulo modification modes. The special-modulo mode enables you to stack memory buffers without the alignment usually associated with data buffers. Both cores can generate four
independent 16-bit memory addresses.

**Special instructions**—The Carmel DSP uses a 24-bit instruction that you can extend to 48 bits for wider operand selection, larger immediate-operand fields, and direct-operand references. Carmel DSP's CLIW architecture extends the traditional DSP instructions into VLIW capability through an additional 96 bits of the CLIW memory. The application-specific CLIW instruction specifies as many as six parallel operations that can use the two ALUs; the two MAC units perform two data moves on the 10XX.

The Carmel DSP family supports conditional execution with the Carmel DSP-predication mechanism to avoid branch penalty and fast context switching with a register-bank-exchange instruction and a conditional execution-load instruction. The register-bank-exchange instruction allows you to specify which registers to shadow. The hardware-looping mechanism enables zero-overhead loops, nested to as many as four levels. The back-trace instructions accelerate the Viterbi-decoder implementation.

All arithmetic/logic instructions support double precision. In addition, special instructions are available for square, division, minimum/maximum, block floating point, logical and arithmetic shifts, bit manipulations, fractional and integer arithmetic, limiting, saturation, and nearest and convergent rounding modes.

**Support**—Infineon provides an integrated program-development environment with uniform interfaces running under Windows. The software tools include an assembler/linker, a C++ compiler, and Tasking's ([www.tasking.com](http://www.tasking.com)) simulator and debugger. The simulator is instruction-, cycle-, bit-, and pipeline-accurate. Algorithm libraries are available as both C and assembly-language routines for common DSP applications and functions. An RTOS facilitates task-level debugging. Infineon provides an evaluation/development board that supports JTAG-based emulation using Carmel's on-chip debugging-support capability. These tools allow you to run programs in real time and within the application's hardware system. You can also check out Infineon's partner section at [www.infineon.com/dsp](http://www.infineon.com/dsp).