Carmel Enables Customizable DSP

User-Defined Instructions, Licensing Plan Set New Siemens Core Apart

MICROPROCESSOR
FORUM 98

*by Jennifer Eyre and Jeff Bier,
Berkeley Design Technology, Inc.*

Earlier this year, Siemens Semi-
conductors announced its new
licensable DSP core known as Carmel. Developed
jointly by Siemens and Israeli design house I.C. Com,
Carmel is a 16-bit fixed-point processor with most of
the standard features found in mainstream DSPs plus
some significant enhancements. At the
recent Microprocessor Forum, Ralph
Sucher of Siemens explained that
Carmel contains two data paths and
can execute up to two 24-bit in-
structions in every clock cycle. In addi-
tion, Carmel has a unique twist: it
allows the user to define VLIW-like
superinstructions, each of which can
specify up to six native instructions to
execute in parallel.

Carmel will operate on a 2.5-V
supply, and Siemens is targeting a typ-
ical power consumption (excluding
peripherals and memory) of 200 mW
at 120 MHz in a 0.25-micron process.
Siemens is betting that the combina-
tion of dual data paths and user-defined instructions
will give Carmel a serious performance advantage
over more traditional DSPs while maintaining low
power consumption. If it achieves its goal, Carmel
should have an energy efficiency (performance per
watt) comparable to that of TI's 'C54x, the most
energy-efficient mainstream DSP available. With this
combination of speed and power consumption,
Siemens seeks a foothold in the lucrative wireless tele-

**Ralph Sucher of Siemens speaks
about the unique CLIW architec-
ture of the Carmel DSP core.**

MICHAEL MUSTACCHI

com market. It is also targeting xDSL, set-top boxes,
and other Internet applications.

Siemens expects Carmel silicon during the first
quarter of 1999; this date has slipped somewhat from
the initial projection of 3Q98. The Siemens roadmap
includes future versions of Carmel with a 24-bit data
width (useful for high-fidelity audio applications, such
as AC-3), as well as a lower performance, lower cost ver-
sion dubbed Tiny-Carmel.

Serious DSP Thrust

Siemens, long a major force in the
electronics industry, is getting into the
DSP business in a big way. In addition
to Carmel, the company has introduced
TriCore, an all-new hybrid DSP/
microcontroller architecture (see MPR
11/17/97, p. 13).

Siemens' business strategy for
Carmel is somewhat unusual, in that
the company will both use the core in
off-the-shelf chips and offer it for
licensing. This approach may be attrac-
tive to OEMs, which can use the chips
during development and initial pro-
duction, then migrate to more cost-
effective core-based ASICs for volume production. It
appears, however, that Siemens does not intend to offer
generic programmable processors based on Carmel;
rather, it will use Carmel exclusively as the DSP engine
for its application-specific chips.

To compete with established architectures such as
TI's 'C54x, Carmel will need a strong application-
development infrastructure. Siemens is the biggest
company to enter the DSP chip market in a long time,

and the company's size and name recognition, combined with the availability of the core for widespread licensing, may help Carmel gain industry acceptance and attract the third-party support that is essential to its success. Beyond the application-development support that any new architecture needs, Siemens must also enable Carmel licensees to integrate the core into their own chip-level products. This requires support for process porting and chip-level integration as well as tools for chip-level verification.

Carmel is the first core Siemens has offered for licensing, and the company now has at least two licensees: I.C. Com *(www.iccomm.com),* which licensed the core back from Siemens, and LSI Logic. In the past, Siemens has used DSP Group's (*www.dspg.com*) OakD-SPCore in its GSM cell-phone handset chips. At least initially, Siemens will continue to use Oak for its existing GSM products while using the more powerful Carmel core for emerging high-end GSM systems and other applications that require more performance than Oak can provide. LSI Logic is also an Oak licensee and may follow a similar strategy.

Carmel is a synthesizable soft core, and Siemens will provide both VHDL and C models. Key blocks, such as the MAC, ALU, shifter, and exponent units, can also be provided in gate-level netlist format to allow optimization in the target fabrication process.

Siemens Semiconductor Spins Off
Carmel's future is also likely to be affected by Siemens Semiconductors' impending spin-off from its parent company, Siemens AG. Once the semiconductor group is an independent entity, it will have to hold its own
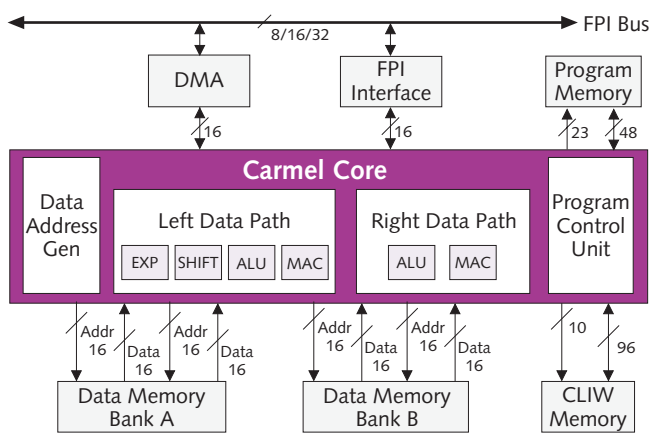

Figure 1. This figure shows a simplified block diagram of the Carmel architecture. Data memory banks A and B can each consist of dual-ported and/or single-ported memory.

against competitors vying for design wins in Siemens AG products. On the other hand, independence may make Carmel more attractive to other OEMs, which won't have to worry about relying on a competitor for their processor. In this sense, Carmel-based chips would have the edge over chips from Lucent and Motorola; in some markets these companies compete with their own customers by selling both the processors and the products that use them.

The spin-off also means that Siemens Semiconductors will no longer have the financial cushion afforded by the larger company; it will have to sink or swim on its own. This need for income may explain, in part, its enthusiastic entrance into the DSP-based IC market, which market-research firm Forward Concepts estimates to be more than $10 billion for 1998 and growing at a rate of 32% per year.

Another VLIW-ish DSP
Lately, it seems that just about everybody is experimenting with VLIW architectures, and Siemens is no exception. Carmel's data paths contain six execution units, and the core can issue and execute up to six native instructions at a time via its user-defined superinstructions. In comparison, Texas Instruments' highly publicized VLIW architecture, the 'C6xxx, has eight execution units and can issue and execute up to eight instructions per cycle (see MPR 2/17/97, p. 14).

It is not possible, however, to determine the two processors' relative performance simply by comparing the number of instructions each can execute in parallel, because of differences in their instruction sets, pipelines, and clock speeds. For example, the 'C6xxx uses simple RISC-like instructions, whereas Carmel's instructions can each perform several operations (e.g., a multiply plus a pointer update).

Carmel's user-defined instructions are referred to as configurable long instruction words (CLIWs). CLIW instructions are like traditional VLIW instructions, in that they combine multiple predefined instructions into a long superinstruction, using position within the superinstruction to route instructions to their intended execution units.

Carmel supports a mixed-width 24- and 48-bit instruction set, and it can execute either one 48-bit instruction or up to two 24-bit instructions at a time. Hence, Carmel essentially has two VLIW-like modes: execution of two predefined instructions at a time or execution of a user-defined CLIW instruction that contains up to six predefined instructions.

Data Paths Not Twins

Carmel's dual data paths are not identical. As Figure 1 shows, one data path contains an exponent unit, a shifter, an ALU, and a MAC unit; the other has only an ALU and a MAC unit. Of the currently available DSPs, Lucent's DSP16xxx (see MPR 9/15/97, p. 11) provides the most similar array of execution units: an ALU, a three-input adder, and a bit-manipulation unit. The two processors share another similarity: both have specialized hardware to support Viterbi decoding, a common function in telecommunications applications. Given this likeness, Carmel-based chips and DSP16xxx parts will probably compete for some of the same telecom sockets.

Carmel's data paths share a common set of registers. Carmel provides a total of six 40-bit accumulators (each providing 8 guard bits) plus 26 address registers, 16 of which can be used as general-purpose registers.

Unlike the 'C6xxx, Carmel is not a load/store architecture; operands can come directly from memory as well as from registers. Some of the registers are shadowed, and a special bank-exchange instruction allows the programmer to swap the contents of a specified list of registers with the contents of their corresponding shadow registers.

Both ALUs operate on 40-bit data. Inputs smaller than 40 bits are zero- or sign-extended, depending on the instruction being executed. In addition to the usual complement of arithmetic and logical operations, the ALUs support division iteration (divide-step) and min/max operations. The ALUs also support SIMD-style addition and subtraction on packed 16-bit data via special add and subtract instructions.

Carmel's MAC units perform single-cycle 17×17-bit multiplications (the extra bit eases implementation of multi-precision arithmetic) on all combinations of signed and unsigned 16-bit operands. The MAC units have the somewhat unusual ability to perform addition and subtraction operations, a feature available only for CLIW instructions.

The exponent unit handles exponent detection and block floating point (a technique for extending dynamic range by asso-ciating one exponent with a block of mantissas). The shifter supports arithmetic and logical shifts by 0–40 bits left or right as well as bit-manipulation operations such as bit-field insert/extract and rotate-through-carry.

Both single- and multiple-instruction loops are supported via a repeat instruction, and hardware loops can be nested up to four levels.

Each data path has its own set of status flags. When 48-bit instructions and 24-bit instructions are executed alone they update one set of the flags; parallel 24-bit instructions and CLIW instructions update both sets.

Mixed Instruction Widths Reduce Code Size

Most of Carmel's instructions are available in both 24- and 48-bit versions. The 24-bit versions support fewer addressing modes and offer less flexibility. For example, 24-bit multiply-accumulate instructions store their results in one of the source operands, whereas the 48-bit version can specify a separate destination register. When code density is critical, the programmer can use the 24-bit instructions; when flexibility is needed, 48-bit instructions are there for the taking.

When coding parallel 24-bit instructions, the programmer specifies one instruction to be executed in the left data path and one to be executed in the right data path. There are restrictions on pairing; for example, instructions that use the exponent unit and shifter cannot execute in parallel, since only the left data path contains these units. Also, two instructions that change program flow cannot execute in parallel. Table 1 provides a summary of Carmel's predefined instructions, along with a description of the pairing restrictions.

Operation	Data Path	Operation	Data Path	Operation	Data Path
Arithmetic		**Multiply**		**Program Flow**	
Add/Subtract	L & R	Multiply-Accumulate	L & R	Branch Abs/Rel	L or R
Add Absolute Value	L & R	Mul-Accum Aligned	L & R	Branch Cond	L or R
Add w/Carry	L & R	Multiply-Subtract	L & R	Break Cond	L or R
Subtract w/Borrow	L & R	Square	L & R	Call Abs/Rel	L or R
Add/Sub High/Low	L & R	Square-Accumulate	L & R	Call Cond	L or R
SIMD Add/Sub	L & R	**Bit Manipulation**		Continue	L or R
Negate	L & R	Change/Set/Clear Bit	L or R	Return Cond	L or R
Round	L & R	Test Bit	L or R	RTI/RTI Cond	L or R
Limit	L & R	Test Field	L & R	Repeat	L or R
Absolute Value	L & R	Compare	L & R	**Miscellaneous**	
Exponent	L	Shift Arith/Logical	L	Leave/Link	L or R
Divide Step	L & R	Insert/Extract	L	Push/Pop	L or R
Min/Max	L & R	Rotate thru Carry	L	NOP	L & R
Increase/Decrease	L & R	**Logical**		Trace/Trap	L or R
Clear Accumulator	L or R	AND/OR	L & R	Bank Exchange	L or R
Clear & Rnd Accum	L or R	NOT/XOR	L & R	Move	L & R

Table 1. When executed as one of a pair, "L" indicates that the instruction can be executed only in the left data path; "L or R" indicates that it can be executed in either data path but not both data paths simultaneously; "L & R" indicates that the instruction can be executed in either data path or both simultaneously.

```
CLIW name (ma1, ma2, ma3, ma4)        ;CLIW reference line
{
MAC1 || ALU1 || MAC2 || ALU2 || MOV1 || MOV2   ;CLIW def.
}
```

Figure 2. Programmers indicate which instruction will be executed in each execution unit, using the instruction's position within the long CLIW instruction. Up to four memory operands can be specified using ma1 through ma4.

Nearly all Carmel instructions (including the arithmetic operations contained within long CLIW instructions) can use predicated execution. Two conditional-execution registers allow programmers to check for two different conditions, giving them control over two parallel 24-bit instructions or two pairs of instructions within a CLIW instruction.

Carmel's 48-bit instructions are fairly orthogonal, while 24-bit instructions have a number of restrictions on register usage and addressing modes. Carmel's instruction set is more orthogonal than typical DSP instruction sets, however, which tend to use highly specialized and cryptic instructions. Carmel's assembly language is C-like and includes several instructions that make life easier for a C compiler, such as push and pop with stack pointer updates.

CLIW Is Key to Performance

Where Carmel differs from nearly every other DSP is in its user-definable instructions. The only other announced DSP with user-definable instructions is the Philips R.E.A.L. DSP core, which predates Carmel and is in volume production.

Typical DSP applications spend most of their time in a few small sections of code. Traditionally, DSPs have

```
a4 = maxnum;
rep (M/2) block
{
  a0l = *(r4) - *(r0++) || a2l = *(r4++) - *(r0+rn0);
  clr (a1, a3) || rep(N-1) single
{
    CLIW VQ1 (r4++, r0++, r0+rn0)
    {
      a0l = *ma1 - *ma2
   || a1 += sqr(a0l)         CLIW Instruction
   || a2l = *ma1 - *ma3
   || a3 += sqr(a2l)
    }
}
  a1 += sqr(a0l) || a3 +=sqr(a2l)
  minm(a4, a1, r0);
  minm(a4, a3, r0+rn0);
}
```

Figure 3. Carmel code example. A CLIW instruction (VQ1) is shown in the shaded box. The "‖" notation is used to designate that instructions are to be executed in parallel.

achieved strong performance by accelerating operations commonly found in these critical sections. Carmel takes this philosophy a step further, allowing application developers to define new instructions to boost performance. Siemens expects CLIW instructions to give the processor a significant bump in performance without incurring the penalties commonly seen in VLIW architectures: i.e., high program memory use (a result of using wide instructions) and high power consumption (a result of using wide on-chip buses).

CLIW instructions are 144 bits long. The first 48 bits are essentially a standard instruction, called the CLIW reference line, that fetches the CLIW instruction definition and up to four memory operands. As Figure 2 shows, the six fields of the CLIW instruction definition encode operations for up to four execution units plus two parallel data moves. Each field can contain one predefined instruction. Both the reference line and the definition must be explicitly specified each time the CLIW instruction is used, as Figure 3 shows. The assembler automatically stores the 48-bit reference line in program memory and stores the 96-bit definition in a separate CLIW memory. The overhead associated with fetching these two components is masked by the processor's eight-stage pipeline and does not adversely affect performance.

Most predefined instructions can be used within CLIW instructions. A few, such as add-and-round, are available for use only within a CLIW instruction. Carmel programmers can freely intermix 24-bit, 48-bit, and CLIW instructions without the use of mode bits.

Unusual Data Capabilities

With the ability to use four execution units at a time comes the need for high memory bandwidth to achieve top performance. Hence, Carmel is equipped with four on-chip 16-bit data buses and a data addressing unit (DAU) that can generate up to four 16-bit addresses and four address modifiers per cycle. This combination allows Carmel to read four 16-bit data words every cycle. In contrast, most of the conventional DSPs, such as the 'C54x and Oak, are limited to only two data words per cycle.

The ability to generate four addresses per cycle is unique and powerful, giving Carmel unusual flexibility in data retrieval. While the DSP16xxx and 'C6xxx share Carmel's ability to retrieve four 16-bit words per cycle, they can generate only two addresses per cycle. Hence, they can achieve their peak memory bandwidth only when data words are arranged as pairs in memory, a restriction Carmel does not impose.

Carmel supports three primary address spaces: program, data, and I/O. Predefined instructions and CLIW reference lines are stored in 48-bit program memory, accessed via a 23-bit address bus. CLIW instruction definitions are stored in a separate 1K × 96-bit memory space. Carmel's data memory is divided into two blocks, each having its own pair of independent buses. Therefore, achieving the maximum data memory bandwidth requires the data to be distributed between the two memory blocks. Each block of data memory can be further divided into dual-ported and single-ported regions.

Carmel supports the addressing modes that are common among DSP chips, including register-direct, memory-direct, memory-indirect with post-modification and index options, modulo addressing (for circular buffering), and bit-reversed addressing. Only 48-bit instructions allow all of the available addressing modes; 24-bit and CLIW instructions support a subset. The program control unit (PCU) is responsible for program address generation, instruction fetch and decode, hardware loops, and pipeline control.

In addition to the data- and program-memory buses, Carmel provides a separate bus that connects the core to peripherals and the DMA unit. The Flexible Peripheral Interface (FPI) bus can be as wide as 32 bits but may be configured to be 8 or 16 bits wide. The FPI bus accesses data memory via Carmel's DMA controller, which can be configured to support from one to eight independent DMA channels. The DMA interface transfers data between peripherals and memory in the background, using cycle stealing.

Moderate Pipeline For Moderate Speed

Carmel's pipeline has eight stages: two instruction-fetch stages, two instruction-decode stages, two data-fetch stages, an execution stage, and a write-back stage. This is deeper than most conventional DSPs, which generally use 3 or 4 stages, but not as deep as the 'C62xx, which uses 11 stages. For instructions that do not affect program flow and do not write to memory, Carmel has single-cycle latencies. This is a significant improvement over the 'C62xx, which incurs a two-cycle latency for multiplies and a five-cycle latency for loads.

Carmel supports delayed branches. Up to three instructions can be executed in the branch delay slots, five if the branch is conditional. The pipeline is not interlocked, so the assembly programmer or compiler must take care to generate results at the correct times.

Siemens is not targeting maximum performance for Carmel; with an eight-stage pipeline and a

0.25-micron process, 120 MHz is not a particularly aggressive goal. Instead, Siemens is trading off speed for easier software development (due to single-cycle latencies), lower power consumption, and easier system-on-a-chip design—an approach that makes sense, given the core's target of high-volume embedded applications.

Zero to Sixty in …

With its dual data paths, VLIW-like multi-issue architecture, and target clock rate of 120 MHz, Carmel will be much faster than any other licensable DSP core currently available. The OakDSPCore, for example, does not have nearly as powerful an architecture as Carmel, and current Oak implementations top out at around 80 MHz. Carmel should also deliver substantially better performance than conventional DSPs with similar clock rates, such as the 'C54x.

Although Carmel will not be able to compete with the speed of the 'C62xx (which TI expects to have running at 250 MHz within the next few months), the two processors are architecturally similar. Carmel and the 'C62xx have comparable arrays of execution units and levels of parallelism, but Carmel has more flexible addressing and lower latencies for most operations. It is therefore likely that Carmel will have cycle counts that are similar to and, in many cases, lower than those of the 'C62xx. And indeed, benchmark results provided by Siemens support this assertion.

Carmel, however, achieves these cycle counts with an on-chip instruction bandwidth of only 144 bits per cycle compared with 256 bits per cycle for the 'C62xx, and Carmel needs only a small amount of wide memory because of its CLIW approach. In part because Carmel doesn't need the wide on-chip buses required by the 'C62xx, it does not incur the high power-consumption penalty so visible on the TI chip. Carmel's projected power consumption is about 10× lower than that of the TI chip despite the fact that the 'C62xx uses a 1.8-V supply, whereas Carmel uses a 2.5-V supply.

Tools Support in the Works

Siemens is collaborating with at least two third parties on tools and software for Carmel. Well-known tools developer Tasking has developed a suite of Windows-based software development tools for Carmel, including an assembler, a linker, and an instruction-set simulator, all of which are available in beta versions. Tasking (*www.tasking.com*) is currently developing a C compiler for Carmel, which Siemens expects to become available in January. In addition, Eonic Systems (*www.eonic.com*) is developing a real-time operating system (RTOS) for Carmel, expected to become available in 2Q99.

Carmel's assembler, linker, and simulator all support CLIW instructions. Siemens states that the C compiler will be able to synthesize CLIW instructions, though probably not in the initial release.

Blazing New Trails

At a time when many semiconductor vendors are opting to license outside cores rather than develop their own, it may seem surprising that Siemens has decided to develop not just one but two new DSP-capable architectures. Siemens is driven by two motivations. First, anticipating the increased importance of DSP in its products, Siemens has a strong desire to control its own architectural destiny rather than be subject to the whims of outside core suppliers. Second, the selection of licensable DSP cores available today is actually quite small, and none matches the performance and energy efficiency projected for Carmel.

Carmel's combination of traditional DSP architecture, VLIW techniques, and user-defined instructions promises excellent efficiency in typical DSP applications. Carmel will serve Siemens well in its own application-specific standard ICs, such as cellular-telephone chip sets. Carmel's success in the broader market is less certain, as it requires gaining acceptance among other semiconductor vendors and large system developers. Winning semiconductor vendor support will be an uphill battle, since competing vendors may be reluctant to depend on Siemens for cores. This may account for Siemens' limited success to date in licensing Carmel. In addition, Siemens must show that it can deliver the tools, support, and other development infrastructure required to enable licensees to quickly develop Carmel-based ICs. This will be no simple task, even with Siemens' substantial resources. Ⓜ

*Authors Jennifer Eyre and Jeff Bier are with Berkeley Design Technology, Inc. (BDTI), the DSP technology analysis and software development firm (*www.bdti.com*).*