Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

FLOAT WITHOUT BLOAT

ARC Adds Economical Floating Point to Customizable Processor Cores

By Tom R. Halfhill {5/23/05-02}

For years, ARC International has considered adding an optional floating-point unit (FPU) to its 32-bit customizable processor cores, but it has always been deterred by the cost of the additional logic gates and power. A fully equipped FPU with its own pipeline and register file could double or triple the silicon area of a small embedded RISC processor.

At last week's **Spring Processor Forum**, ARC unveiled FPX—Floating-Point eXtensions—which significantly improve on the performance of a software-emulation library while requiring fewer gates than a complete FPU. ARC's solution is remarkably similar to the optional floating-point extensions for MicroBlaze v4.00 that Xilinx announced at the same session of SPF. One important difference is that ARC's extensions support both single- and double-precision operations, whereas the Xilinx extensions are limited to single-precision operations. (See *MPR 5/17/05-02*, "MicroBlaze Can Float.")

FPX is an extra-cost option for the ARC 600 and ARC 700, the company's licensable intellectual-property (IP) processor cores. The ARC 600 is designed primarily for low-power, deeply embedded applications that need 32-bit processing. (See *MPR 12/15/03-01*, "ARC Alters Trajectory.") The ARC 700, the company's flagship product, is a more powerful processor for higher-end embedded systems, yet it still consumes relatively little power. (See *MPR 6/21/04-01*, "ARC 700 Secrets Revealed.") Both cores are highly configurable, allowing users to add custom instructions, DSP extensions, registers, and other application-specific features. FPX is simply another extension package available in *ARChitect*, the company's graphical configuration tool.

Single-precision FPX is available for licensing now. ARC says the double-precision extensions will be available early this summer. Both packages are supported by ARC's MetaWare development tools, including the C/C++ compiler, assembler, debugger, instruction-set simulator, and cycle-accurate simulator. Two new compiler flags (–Xspfp for single precision and –Xdpfp for double precision) automatically invoke the new instructions—no assembly required.

Extensions Derived From Application Profiling

ARC's engineering team in Elstree, England, began the project by profiling some real-world embedded applications that use ARC's floating-point software library. In particular, ARC profiled some customer programs that calculate coordinates for Global Positioning Systems (GPS). Performance profiling is an important first step in defining any custom extensions for a configurable processor, and ARC's MetaWare development tools include a sophisticated profiler utility. These tests revealed that only two function calls—addition and multiplication—accounted for 60% of the clock cycles in math-intensive floating-point code. Division accounted for 14% of the cycles, and exponentiation was responsible for 5%. All other types of function calls were less significant.

After identifying the low-hanging fruit, ARC decided to pluck it by designing a few extension instructions that would accelerate basic floating-point operations without requiring a full-blown FPU. As a result, FPX is not an FPU coprocessor with its own instruction pipeline and register file. The new instructions share the same pipeline with integer instructions and use the same general-purpose integer registers. (To support double-precision instructions, FPX adds four auxiliary registers.) In all, FPX adds only seven new instructions, not counting variations of arithmetic instructions that specify different sources for operands and so forth. ARC's conservative approach saves thousands of gates, keeps the processor core small and power efficient, and removes the need to communicate with a coprocessor.

In those respects, ARC is even thriftier than Xilinx, which recently added 10 floating-point instructions to its synthesizable 32-bit processor core, MicroBlaze v4.00. Both companies chose to keep their RISC cores small by allowing integer and floating-point instructions to share the same pipeline and registers. Of course, the trade-off is performance. Floating-point instructions, which typically have longer execution latencies than integer instructions have, can stall the uniscalar pipeline in these processors, preventing even nondependent integer instructions from bypassing and executing. Also, the floating-point operands can occupy the integer registers like unwelcome squatters, sometimes creating resource dependencies that impair performance. Nevertheless, we think ARC and Xilinx made worthwhile trade-offs, because their new instructions are much faster than software floating-point routines and require less silicon than a full FPU might have.

Despite adding only seven new instructions, FPX supports single- and double-precision math. The SP FPX package has three instructions for 32-bit addition, subtraction, and multiplication. The DP FPX package has four instructions: 64-bit versions of the same basic arithmetic operations plus a special register-copy instruction. Each package is optional, so customers can add only the extensions they need by clicking a few buttons in *ARChitect*. Table 1 lists all the new instructions with descriptions, execution latencies, and comments.

All the new instructions are straightforward and self-explanatory, save one: dexcl, a special register-exchange instruction. FPX needs dexcl because ARC didn't add any 64-bit-wide registers to the processor for double-precision operands or results. Instead, the 64-bit values occupy pairs of existing 32-bit integer registers and pairs of new 32-bit auxiliary registers. ARC's clever workaround conserves gates but requires some explanation.

By default, an ARC processor has a 32-entry file of 32-bit integer registers, also known as core registers. Auxiliary registers are optional. Users can add several core registers and a virtually unlimited number of 32-bit auxiliary registers, which programs can access with special single-cycle load/store instructions. The DP FPX package defines four new auxiliary registers for storing 64-bit input operands and results.

A double-precision instruction that needs two 64-bit input operands retrieves one of them from a pair of 32-bit core registers and the other from a pair of new 32-bit auxiliary registers. The dexcl instruction copies two core registers to a pair of auxiliary registers in a single-cycle operation, forming one of the 64-bit operands. (If a previous double-precision instruction leaves its result in a pair of auxiliary registers, and the following arithmetic instruction uses that result as an input operand, the program doesn't need dexcl for this purpose.) An arithmetic instruction always stores its 64-bit result in a pair of auxiliary registers and also copies the most significant 32 bits of the result to a destination core register. If the program needs to manipulate the entire 64-bit result using core registers, the dexcl instruction also copies the lower 32 bits of the result to a core register.

Floating-Point Performance Is Much Improved

FPX complies with parts of the IEEE 754 floating-point standard that are most useful in deeply embedded applications. The new instructions handle signed-zero and infinity operations, denormalized numbers, and "quiet" not-a-number (NaN) exceptions. They also support the IEEE 754 round-to-nearest mode but not the up- or down-rounding modes. Signaling NaNs are handled as quiet NaNs, not as exceptions, because FPX is designed to work with both the ARC 600 and ARC 700 processors, and the ARC 600 doesn't support precise exceptions. The processor calls the usual floating-point software libraries to handle any situations that FPX doesn't support in hardware.

Performance improvements with FPX can be significant. The latency column in Table 1 shows the number of clock cycles each instruction can save when substituted for equivalent functions in the software floating-point library. After profiling the library and some real-world applications from three customers, ARC found that the SP FPX package improved floating-point performance by an average factor of 2.27×. In the same tests, the DP FPX package accelerated performance by 3× to 7×. Of course, the actual speedup for a particular application depends on which operations it performs and how much time it spends in those routines. Figure 1 compares the number of clock cycles required for various

Instruction	Description	Latency	Notes
Single-Precision Floating Point			
fadd	FP add	3 cycles	Replaces 70-cycle software library function
fmul	FP multiply	3 cycles	Replaces 40-cycle software library function
fsub	FP subtract	3 cycles	Replaces 70-cycle software library function
Double-Precision Floating Point			
daddh	FP add	6 cycles*	Replaces 95-cycle software library function
dexcl	Copy registers	1 cycle	Copy two 32-bit core registers to auxiliary registers to form a 64-bit input operand, and copy lower 32 bits of result from an auxiliary register to a destination register in the core register file
dmulh	FP multiply	4 cycles*	Replaces 60-cycle software library function
dsubh	FP subtract	4 cycles*	Replaces 95-cycle software library function

Table 1. ARC's FPX has separate packages of floating-point extensions for single- and double-precision operations. In all, there are only seven new instructions. The single-precision instructions require three clock cycles to execute but can be pipelined for single-cycle throughput. *In ARC's preliminary design, double-precision instructions execute in four cycles.

single-precision operations using FPX and the software floating-point library.

What is the price for these improvements? The SP FPX package requires about 12,000 to 14,000 NAND-equivalent gates, and the DP FPX package requires about 24,000 to 30,000 gates. Together, they would require 44,000 gates, but ARC says no customers so far have expressed interest in using both packages in a single processor. The DP package is larger both because of the extra complexity of the instructions and because of the auxiliary registers needed for 64-bit data.

Putting these gate counts in perspective: the base configuration of the ARC 600 (excluding caches and 32-bit integer multiplier) is about 27,000 gates, whereas the base configuration of the ARC 700 is about 100,000 gates. Although adding the DP FPX package to the ARC 600 would double the base size of the core, the inflation is less significant in a typical processor configuration, which would probably include caches and other extensions. In actual silicon using a deep-submicron fabrication process, the extra size is negligible.

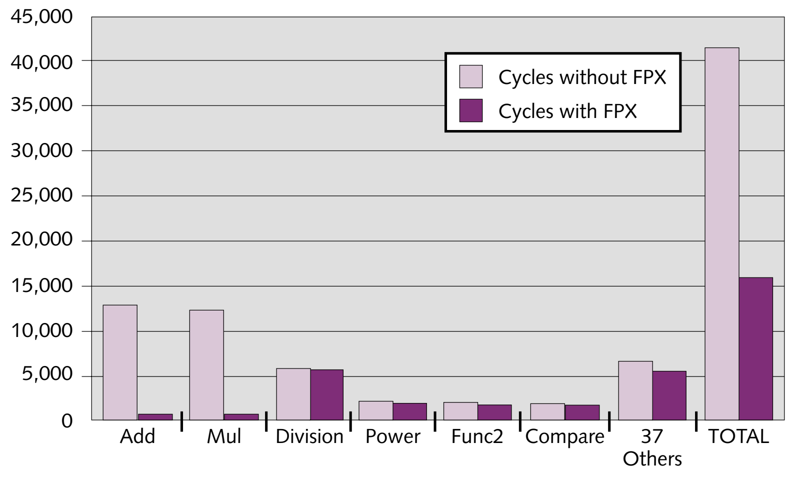

Figure 1. In ARC's tests with a customer's GPS application, floating-point addition and multiplication showed the greatest improvements when using single-precision FPX instructions instead of software routines. In this test, there was no improvement for division (FPX lacks division instructions) and only scant improvements with other operations. ("Func2" is an undisclosed application-specific function in the customer's program.) Overall, the code using FPX instructions ran more than twice as fast.

Customers needing more floating-point performance than FPX provides still have the option of creating additional extensions with ARC's Extension Instruction Automation (EIA) tools. Indeed, ARC created FPX using those tools. For a quick fix, however, FPX provides significantly better performance than the software libraries do—without busting the budgets for silicon or power.

Competitors Have Stronger FPUs

ARC's closest competitors for licensable 32-bit embedded processors are ARM, MIPS Technologies, and Tensilica. All have FPUs that outclass ARC's FPX in some ways, although their FPUs tend to be larger and consume more power. One exception is Tensilica's single-precision FPU, which is scarcely larger than ARC's more limited single-precision extensions.

Cambridge Consultants, which introduced a new licensable 32-bit processor core at SPF, doesn't currently offer a hardware floating-point option. Altera and Xilinx have licensable 32-bit processors, but their cores are primarily intended for FPGAs, not for conventional SoCs, and only the new Xilinx MicroBlaze v4.00 has floating-point logic. Table 2 summarizes the options available from ARC, ARM, MIPS, and Tensilica.

Feature	ARC FPX	ARM VFP9-S	ARM VFP10 / VFP11	ARM VFPv3	MIPS MIPS32	MIPS MIPS64	Tensilica Xtensa FPU
CPU Cores	ARC 600, ARC 700	ARM9 family	ARM10 family ARM11 family	ARM11 family	MIPS32 24Kf	MIPS64 5Kf, 20Kc	Xtensa III, IV, V, LX
Synthesizable	Yes	Yes	VFP11	n/a	Yes	5Kf	Yes
Hard Macro	—	—	VFP10	n/a	—	20Kc	—
FP Precision	32 or 64 bits	32 / 64 bits	32 / 64 bits	32 / 64 bits	32 / 64 bits	32 / 64 bits	32 bits
FP Pipeline	—	Yes	Yes	Yes	Yes	Yes	Yes
FP Registers	2 x 64 bits*	32 x 32 bits or 16 x 64 bits	32 x 32 bits or 16 x 64 bits	32 x 64 bits or 128 x 16 bits	32 x 32 bits or 16 x 64 bits	32 x 64 bits or 64 x 32 bits [†]	16 x 32 bits
Size	12KG–14KG (SP) 24KG–30KG (DP)	100KG–130KG	1.16mm^2 0.13µ (VFP10)	n/a	~4.5mm^2 (CPU + FPU)	n/a	17.4KG–25KG
Peak SP MFLOPS / MHz	1.0	1.3	2.0	n/a	n/a	n/a	2.0
Availability	Now (SP) Summer (DP)	Now	Now	Now	Now	Now	Now

Table 2. With ARC's introduction of FPX, all the major processor-IP vendors now offer floating-point options for their embedded RISC cores. However, ARC's competitors have FPUs with independent pipelines and register files, whereas FPX instructions share the existing integer pipeline and registers. Consequently, ARC's performance suffers in comparison, but its single-precision extensions require fewer gates than do competing solutions. *FPX adds four 32-bit auxiliary registers; each pair of registers can hold a 64-bit operand or result. [†]The MIPS64 FPU can pack pairs of 32-bit values into 64-bit registers for parallel operations. n/a: information not available.

Price & Availability

ARC's FPX (Floating-Point eXtensions) for the ARC 600 and ARC 700 processors are synthesizable Verilog models. The single-precision FPX package is available now; the double-precision extensions are scheduled for release early this summer. Both extension packages are extra-cost options in the *ARChitect-2* graphical processor-configuration tool included with ARC processors. ARC hasn't publicly disclosed the licensing fees for FPX. For more information, see *www.arc.com/newsandevents/PressRelease.html?id=197*.

ARM's current FPUs are optional coprocessors based on the VFPv2 (Vector Floating-Point version 2) architecture, and ARM recently announced VFPv3 for future implementations. The existing VFPv2 supports single- and double-precision math, fused MAC (FMAC) instructions, and SIMD instructions for applications requiring high data throughput, such as 3D graphics. These FPUs are independently pipelined and have their own register files, which are visible to programmers as 32 single-precision or 16 double-precision registers. The FPUs also support the most important IEEE 754 standards in hardware, although some functions require library calls.

VFP9-S is a synthesizable FPU for ARM9E processors. It can execute a basic floating-point operation every two cycles. However, by using its parallel load/store instructions, vector-processing mode, and division/square-root engine, the VFP9-S can deliver a peak 1.3MFLOPS per megahertz. Depending on the way it's optimized during synthesis, VFP9-S requires 100,000 to 130,000 gates, making it quite a bit larger than ARC's FPX, although it's also more capable.

VFP10 is a hard macro for ARM10E processors. It has the same capabilities as the VFP9-S, but it can issue a VFP instruction on each clock cycle and transfer data over dual 64-bit load and store buses, giving it peak performance of 2MFLOPS per megahertz. VFP10 occupies about the same silicon area as VFP9-S does in the same fabrication process, because it's an optimized hard macro instead of a synthesizable model. ARM estimates VFP10 occupies about 1.16mm^2 in TSMC's 0.13-micron LV process.

VFP11 is a synthesizable FPU that brings all the features of VFPv2 to ARM11 processors. It's designed to run at the same clock speeds as ARM11 cores, and it provides the same performance as VFP10.

Less is known about ARM's latest floating-point architecture, VFPv3, announced last year for the ARMv7 architecture. It doubles the size of the register file, with 32 double-precision registers also visible as 16 128-bit registers. VFPv3 has some configurable fixed-point conversion instructions, and user exceptions are optional—the FPU can run trap-free at top speed, providing the IEEE-specified results for exceptional conditions. FPUs based on VFPv3 will probably claim more silicon than ARM's existing FPUs, if only because of the larger register file.

MIPS processors offer powerful floating-point capabilities, too—not surprising for a seminal RISC architecture originally designed for workstations. FPUs supporting single- and double-precision floating point are optional in both the MIPS32 and MIPS64 embedded-processor architectures. These FPUs have dedicated pipelines, registers, and condition codes. In MIPS32, the FPU register file is visible as 32 32-bit registers or 16 64-bit registers. In MIPS64, the FPU has 32 64-bit registers and optional single-precision SIMD instructions that pack two 32-bit operands into a single 64-bit register. FPUs are standard equipment in the synthesizable MIPS32 24Kf and MIPS64 5Kf processors as well as in the hard-core MIPS64 20Kc.

Tensilica's optional FPU, introduced in 2000 for Xtensa III, is also available for the latest Xtensa LX. Although this FPU is limited to single precision, it's a true coprocessor that has its own pipeline and 16-entry register file. It adds 34 instructions, including special load/store instructions, plus some offset and indexed address-update modes. Most arithmetic operations (addition, subtraction, multiplication, multiply-add, and multiply-subtract) have four-cycle latencies—one cycle longer than ARC's single-precision FPX instructions. Loads and data-conversion instructions execute in two cycles, and moves and compares execute in one cycle. By pipelining the fused instructions, Tensilica's FPU can sustain 2MFLOPS per megahertz. Remarkably, this pipelined FPU with its own register file adds only 17,400–25,000 gates to the processor core, depending on whether it's synthesized for area or speed.

FPX Achieves ARC's Goals

Although FPX is less capable than some competing FPUs, it meets ARC's goal of float without bloat—improving floating-point performance on the ARC 600 and ARC 700 without sinking the diminutive cores under a burden of extra silicon. Some ARC 600 users may pause before clicking on the FPX option in *ARChitect*, but it's almost a no-brainer for the ARC 700 in applications requiring faster performance. In addition, FPX provides a head start for customers developing their own floating-point extensions for a specific application.

Another important feature easily overlooked is that the SP and DP FPX packages are independent—customers can add only the level of floating-point precision they need. With other architectures, the floating-point extensions are either limited to single precision or include both single precision and double precision, inflating the gate count. ARC's rolling stone gathers no CMOS. ◇