Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

ARM THUMBS A RIDE

New Cortex-R4F Processor Adds FPU and ECC for Automotive Market

By Tom R. Halfhill {10/30/06-01}

On average, there are 1.3 ARM processor cores per cellphone. And these days, it seems as if half the motorists on the road are yapping on their cellphones while driving. So, in a way, ARM already has a strong presence in the automotive market—though not exactly in the way the company desires.

ARM wants to see more of its processors built *into* automobiles, not merely *used* in automobiles. As Figure 1 shows, the market for embedded processors in motor vehicles is large and growing—some luxury cars have as many as 80 chips. ARM already has automotive design wins at Analog Devices, Freescale, Micronas, NXP Semiconductors (formerly Philips Semiconductors), Oki Semiconductor, Sharp, STMicroelectronics, and Texas Instruments. ARM-based chips from those companies are controlling power trains, antilock brakes, airbags, sensors, dashboard instruments, and entertainment systems. Ironically, it might be an ARM-based airbag controller that saves the lives of drivers who were too distracted by their ARM-based cellphones.

Today, ARM's automotive design wins are based on older cores, such as the ARM7TDMI and ARM966. Newer designs need more processing power. So on October 9, at **Fall Microprocessor Forum** in San Jose, California, ARM announced the Cortex-R4F licensable processor core, an improved version of the Cortex-R4 unveiled at Spring Processor Forum last May. (See *MPR 5/16/06-01*, "ARM Reveals Cortex-R4.") ARM is steering the Cortex-R4F directly into the automotive market. The 32-bit synthesizable processor is also suitable for other embedded-control applications that need floating-point arithmetic and high reliability.

The "F" appended to the Cortex-R4F's product name stands for both FPU and fault tolerance. Specifically, ARM has enhanced the Cortex-R4F with a 32/64-bit FPU (IEEE 754 compliant) and internal error-correction codes (ECC) on all local memory. In contrast, the existing Cortex-R4 lacks an FPU and requires developers to implement their own external ECC logic, although the processor does support simple parity protection for local memory. The new features (as well as some others) are optional, so developers may configure the Cortex-R4F in various ways for different applications.

In addition, the Cortex-R4F is slightly more power efficient than its immediate predecessor. According to ARM's

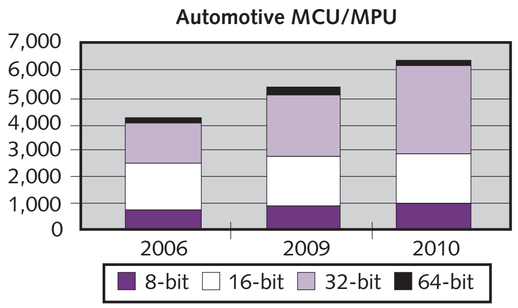


Figure 1. This six-year forecast anticipates strong growth for semiconductors in automotive systems, especially for 32-bit processors, which can handle the additional processing demands of fuel-efficient vehicles and hybrid gasoline/electric engines. Semiconductor revenue is predicted to grow at a faster rate than overall automotive market growth, thanks to more electronics going into vehicles. Another trend is the consolidation of multiple functions within fewer controllers, which reduces the vehicle's chip count and network wiring. (Source: Strategy Analytics)

simulations, the Cortex-R4F will use 5–10% less power than a similarly configured Cortex-R4. When the Cortex-R4F is optimized for speed in a generic 90nm CMOS fabrication process, the maximum clock frequency is 400MHz (worst case, –40°C to +125°C), and power consumption is about 0.317mW per megahertz. In typical automotive applications, ARM expects developers will aim for a lower clock frequency (perhaps 250MHz), which will reduce the processor's size and power. ARM is licensing the Cortex-R4F now and expects to deliver the Verilog models in 1Q07.

New FPU Is Tightly Integrated With the Core

Automotive controllers need floating-point arithmetic for its greater precision, dynamic range, and protection against data overflows. Many automotive programmers write their code in C or use high-level modeling tools like Matlab, which can automatically generate C code that uses floating-point datatypes. In general, single-precision (32-bit) floating-point math is sufficient for these applications, so ARM has optimized the Cortex-R4F for 32-bit throughput. However, the FPU can also execute double-precision (64-bit) operations, albeit at a slower rate.

ARM has offered optional FPUs for its 32-bit processor cores for several years. These include the VFP9-S for the ARM9 family, the VFP10 for the ARM10 family, and the VFP11 for the ARM11 family. All are 32/64-bit floating-point coprocessors with vector-processing instructions, and they connect to the core's coprocessor bus. Their microarchitectures are basically the same. They are independently pipelined coprocessors with register files that provide either 32×32-bit registers or 16×64-bit registers. The VFP9-S and VFP11 are synthesizable cores, and the VFP10 is a hard macro.

The FPU for the Cortex-R4F is a new implementation of the latest VFPv3 vector floating-point architecture, which supports the ARMv7 instruction-set architecture (ISA). VFPv3 differs in an important way from ARM's previous FPUs: it's not a coprocessor that must interact with the host core over the coprocessor bus. Instead, it's tightly integrated with the host core. Indeed, in the Cortex-R4F implementation, the FPU shares the first four stages of the eight-stage integer pipeline. This tighter integration should improve performance by eliminating slow transactions over the coprocessor interface.

Table 1 compares the raw floating-point performance of the Cortex-R4F to that of the ARM VFP11 FPU, Freescale's Power Architecture e200 processor core, and Infineon's TriCore 1.3 processor core. The Power e200 is very popular in automotive controllers, especially in the U.S. and Europe. Neither ARM nor Freescale has published EEMBC scores for these processors, so this table compares the issue latencies and stall delays for typical floating-point operations, as measured in clock cycles.

Frankly, there's not enough information in this table to declare a winner, although VFPv3 is definitely an improvement over VFP11. EEMBC benchmarks would be more welcome. EEMBC's automotive suite is the consortium's largest benchmark suite, comprising 16 separate tests. (See *MPR 5/1/00-02*, "EEMBC Releases First Benchmarks," and *MPR 6/21/99-01*, "Embedded Benchmarks Grow Up.") However, one shortcoming is that the suite doesn't test the speed at which a processor can service the peripherals needed by automotive microcontrollers. EEMBC says it's working on a solution.

Superscalar Execution With Less Overhead

Under ideal conditions, the Cortex-R4F can achieve two-way superscalar execution. Although it has only one ALU, it can sometimes execute two integer instructions—such as a load and an ADD—simultaneously. If there are no data dependencies, an integer operation and a floating-point operation can issue at the same time. Yet the Cortex-R4F isn't an end-to-end superscalar processor, so it needs less control logic than does a fully multipipelined processor like the Cortex-A8. (See our

Floating-Point Operation	ARM Cortex-R4F		ARM VFP11		Freescale Power e200		Infineon TriCore 1.3	
	Latency	Issue Stalls	Latency	Issue Stalls	Latency	Issue Stalls	Latency	Issue Stalls
ADD/SUB	3	0	7 or 8	0	3	0	2	1
CMP	1	0	4	0	3	0	1	0
DIV	16	0 14 for DIV	19	0 14 for DIV	12	12	15	14
Convert to/from Float	3	0	7 or 8	0	3	0	2	1
MUL–ADD/SUB	6	0 or 1	7 or 8	0	3	0	3	2
MUL	3	0	7 or 8	0	3	0	2	1
SQRT	16	0 14 for SQRT	19	0 14 for SQRT	Not supported		Square-root steps only	
Int/FP Move	1 or 2	0	2 or 4	0	Not needed*		n/a	n/a

Table 1. This table shows the number of clock cycles required to perform some common single-precision floating-point operations on the new ARM Cortex-R4F, ARM's older VFP11 FPU, Freescale's Power e200 processor core, and Infineon's TriCore 1.3 processor core. In this comparison, latency is the number of cycles between issuing an instruction and then issuing a data-dependent instruction. The second column under each processor shows the number of cycles between issuing an instruction and then issuing another instruction that doesn't have a data dependency. *Freescale's Power e200 doesn't need to move operands from integer registers to floating-point registers, because it has a shared register file. (n/a: data not available)

two-part coverage in *MPR 10/25/05-02* and *MPR 11/14/05-01*, "Cortex-A8: High Speed, Low Power.")

Figure 2 shows the way ARM appended the Cortex-R4F's floating-point pipeline to the existing Cortex-R4 eight-stage integer pipeline. Not shown are the first four pipe stages, because they are the same for both processors. (Our previously cited Cortex-R4 article has a complete pipeline diagram.) After decoding an instruction, the Cortex-R4F issues it to the integer units or one of the four floating-point pipelines. To reduce the load-use penalty, the floating-point pipes are offset by one clock cycle from the integer pipes, so a floating-point instruction won't need its source data until one cycle after a previous load.

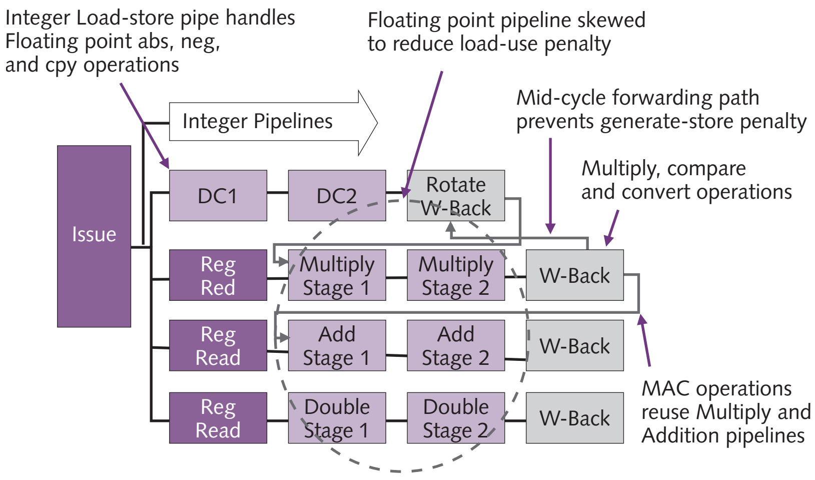

Integer Load-store pipe handles Floating point abs, neg, and cpy operations
Floating point pipeline skewed to reduce load-use penalty
Mid-cycle forwarding path prevents generate-store penalty
Multiply, compare and convert operations
Integer Pipelines
Issue
DC1
DC2
Rotate W-Back
Reg Red
Multiply Stage 1
Multiply Stage 2
W-Back
Reg Read
Add Stage 1
Add Stage 2
W-Back
Reg Read
Double Stage 1
Double Stage 2
W-Back
MAC operations reuse Multiply and Addition pipelines

Figure 2. For efficiency, the Cortex-R4F integrates the FPU pipeline with the existing eight-stage integer pipeline. Under ideal conditions, the Cortex-R4F can execute both an integer and a floating-point instruction simultaneously. This diagram shows only the back end of the integrated pipeline, because the first four stages—fetch1, fetch2, predecode, and decode—are identical. Likewise, a single arrow summarizes most of the integer pipelines. The floating-point pipeline actually has four pipes for handling different kinds of floating-point instructions. At the top, the integer load/store pipe handles floating-point absolute, negative, and copy operations. Dedicated pipes execute other floating-point instructions. Multiply-accumulate (MAC) instructions reuse the multiply and addition pipes.

A typical example of a dual-issue opportunity is a conditional-branch sequence that compares two floating-point values (FCMP), copies the floating-point flags to the status register (FMSTAT), and then branches if the status-register flags meet a particular condition. The Cortex-R4F issues the FCMP instruction first, then issues the FMSTAT and conditional-branch instructions simultaneously. Thanks to pipeline skewing, the FCMP instruction in the floating-point pipeline can forward its result to the branch instruction, which is executing in the integer pipeline one stage behind FCMP. Figure 3 illustrates this sequence, which executes as quickly as an integer compare and branch.

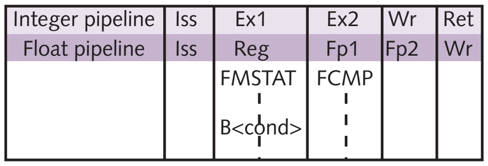
In addition to this dual-issue integer/floating-point capability, the Cortex-R4F can sometimes issue pairs of integer instructions simultaneously, just as the Cortex-R4 can. Multiple integer pipelines following the instruction-issue stage make it possible—again, without the overhead of complex control logic that a fully superscalar processor requires. This bit of clever pipelining allows the Cortex-R4F to deliver more throughput in both the integer and floating-point domains while saving gates and conserving power.

ECC Enables Fault-Tolerant Designs

Besides the new FPU, the Cortex-R4F's other major improvement over the Cortex-R4 is integrated ECC logic. Although the older processor supports both parity checking and ECC on all local memory, only parity checking is included with the core; developers must implement ECC using their own external logic. The Cortex-R4F eliminates that requirement by integrating support for both types of error checking—developers need set aside only some extra memory for the error-correction codes. ECC is an important feature for the automotive market, where fault tolerance and protection against soft errors are vital for power-train controllers.

Parity checking and ECC are individual options for the instruction cache and data cache, as well as for tightly coupled memory (TCM), more popularly known as scratchpad memory. ARM's ECC supports single-error correction (SEC) and double-error detection (DED) for any of those local memories. ARM says the most efficient way to implement error correction in the Cortex-R4F is to protect 64-bit double words, not 32-bit words or smaller data chunks. ECC adds eight bits of overhead to every 64 bits of data. This scheme will detect one or two errors in any position within the double word, and it will correct any single error within the double word. However, ARM allows developers to implement ECC on 32-bit words if they choose—it's a synthesis option. Word-level error correction can detect and correct more errors, at the cost of additional memory overhead for ECC codes.

Figure 4 illustrates the trade-offs. At one extreme, ARM could have allowed developers to apply ECC to individual bytes (the smallest amount of data an ARM processor can read or write is eight bits). The memory overhead for this



Integer pipeline	Iss	Ex1	Ex2	Wr	Ret
Float pipeline	Iss	Reg	Fp1	Fp2	Wr
		FMSTAT	FCMP		
		B<cond>			

Figure 3. Partial superscalar pipelining allows the Cortex-R4F to dual-issue some pairs of integer and floating-point instructions. In this example, the floating-point compare (FCMP) instruction in the second execution stage (Fp2) of the floating-point pipeline is forwarding its result to the conditional branch instruction, which is following one stage later (Ex2) in the integer pipeline. This allows the processor to dual-issue the branch instruction with the instruction that copies the floating-point flags to the status register (FMSTAT).

Price & Availability

ARM is licensing the Cortex-R4F now and plans to deliver the synthesizable Verilog models in 1Q07. ARM doesn't publicly disclose licensing terms or royalties. For more information about the Cortex-R4F and related processors, visit *www.arm.com/products/CPUs/families/CortexFamily.html.*

granularity would be 40 bits of ECC for every eight bytes (63%). At the other extreme is double-word granularity, which requires only eight bits of ECC for every eight bytes (12%).

Note that the double-word ECC scheme doesn't preclude writing smaller chunks of data to memory. A program can still write data in 8-, 16-, or 32-bit widths. However, the Cortex-R4F will carry out those writes by performing a read/modify/write operation on the entire 64-bit double word. (This operation isn't atomic, so another event can interrupt it.) It's also worth noting that developers can synthesize the Cortex-R4F without ARM's ECC in favor of their own error-correction scheme in external logic, as before.

The Cortex-R4F doesn't support parity or ECC protection for local registers, but it's not an oversight. At today's geometries, soft errors are much less of a problem in logic than they are in densely packed memory structures. To detect and recover from hard errors, some developers implement their own fault-monitoring blocks or even a second processor core that mirrors the operation of the primary core. Error protection for register files is a relatively new feature in some server and mainframe processors, such as Fujitsu's SPARC64 V/VI and IBM's POWER6, which were also presented at MPF. (See *MPR 10/23/06-01*, "SPARC64 VI: Ready for PrimeTime," and *MPR 10/30/06-02*, "POWER: The Sixth Generation.")

Developers Can Optimize for Size or Speed

Configuration options allow developers to synthesize the Cortex-R4F without caches, TCMs, the FPU, the memory-protection unit (MPU), or an AMBA-3 AXI slave interface. The configurable instruction and data caches can be as large as 64KB each, and TCMs can be as large as 8MB (far larger than is practical with existing fabrication technology). As we noted in our previously cited Cortex-R4 article, ARM is gradually moving toward greater configurability with its new processors. However, ARM stops short of permitting developers to alter the ISA, unlike ARC International, MIPS Technologies, and Tensilica.

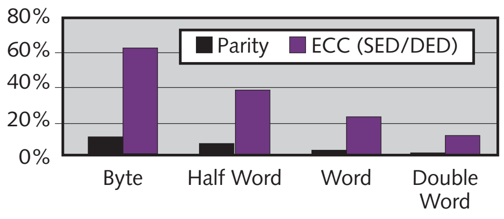


Figure 4. At synthesis time, developers can choose the granularity of error detection and correction in the Cortex-R4F. The processor can protect data widths of 32 bits or 64 bits. Protecting 64-bit double words is the most memory-efficient scheme, reducing the overhead to only 12%.

Table 2 shows the way different configurations and synthesis options can dramatically affect the speed and size of the Cortex-R4F. Hundreds of combinations are possible. This table compares only two configurations, varying the amount of cache, the FPU option, and the synthesis scripts.

Another improvement the Cortex-R4F inherits from the Cortex-R4 is faster interrupt response, a vital feature for automotive controllers. Both processors can respond to an interrupt in 20 clock cycles or fewer, compared with 54 cycles for the ARM966E-S and 118 cycles for the ARM946E-S. The Cortex-R4F's response time compares favorably with other automotive processors, even those that run at much lower clock frequencies.

ARM cites several features that help the Cortex-R4F handle interrupts more promptly. Some features appear in earlier ARM processors, particularly the ARM11 family, but the Cortex-R4 and Cortex-R4F are the first ARM processors having all the features. First, the memory system is nonblocking, so the processor can continue filling cache lines and performing other memory operations while responding to an interrupt. Second, the processor can abort multiword instructions and restart them later. Third, developers can configure what ARM calls a "fast interrupt request" (FIQ) as a nonmaskable interrupt (NMI). (For details, see *MPR 2/21/06-01*, "Can ARM Beat the Clock?")

Fourth, the AXI bus supports out-of-order transactions, so memory operations in an interrupt-service routine can finish executing ahead of previously issued loads and stores. Fifth, the ARMv6 ISA added new instructions for accelerating context switches, such as a single instruction that saves the return state. And finally, the Cortex-R4F has a VIC (vectored

Cortex-R4F Configuration	Physical-IP Libraries (Logic and Memory)	Speed (Worst Case)	Area (Without Cache)	Area (With Cache)	Power
16K caches, FPU (Optimized for speed)	Sage-X Advantage RAM	400MHz	1.53mm^2	2.26mm^2	0.317mW/MHz
8K caches, no FPU (Optimized for area)	Sage-X Advantage RAM	273MHz	0.86mm^2	1.43mm^2	0.244mW/MHz

Table 2. Small differences in configurations and synthesis scripts can have a great effect on the size and speed of the ARM Cortex-R4F, even in the same fabrication process. In this comparison of only two configurations with relatively minor differences, the maximum worst-case clock frequency and die area vary by almost two to one. ARM synthesized both configurations for TSMC's generic 90nm CMOS process.

interrupt controller) port that allows the processor to branch directly to a specific interrupt handler instead of waiting for the software to handle the vector.

Similar Licensable Cores Lack ECC

The Cortex-R4F is suitable for other embedded applications besides automotives. It will compete directly against licensable 32-bit processor cores from ARC, MIPS, and Tensilica. Similar processors with FPUs and either an MPU or a full-fledged memory-management unit (MMU) include the ARC 625D, ARC 750D, all members of the MIPS32 24Kf and 34Kf families, and Tensilica Xtensa 6. All are synthesizable cores introduced in the past two years.

ARC's FPX extension is a fairly recent introduction that is less powerful than the ARM VFPv3. But it's a very small FPU that optionally supports 32- or 64-bit floating-point math, with partial IEEE 754 compliance. (See *MPR 5/23/05-02*, "Float Without Bloat.")

MIPS is known for having the most powerful floating-point capabilities among licensable processor cores—a benefit of the company's previous experience with high-end workstations and servers. The MIPS32 24Kf is the floating-point version of the 24K core. It also has an MMU, so it can run a more sophisticated operating system, such as Windows CE. Another strength of the 24K core is that developers can easily implement multiple register files, permitting rapid context switches. (See *MPR 10/20/03-03*, "MIPS Reveals 24K Core Family.")

Tensilica's optional FPU for Xtensa 6 is the least powerful FPU in this group, because it supports only single-precision operations and doesn't support some IEEE 754 error flags. An MMU is optional. (See the sidebar, "Tensilica Introduces Xtensa 6 Processor Core" in *MPR 11/28/05-01*, "Tensilica Previews Video Engine.") However, *MPR* expects Tensilica will soon announce two new processor cores, the Xtensa LX2 and Xtensa 7, which will have ECC and other enhanced features. Table 3 summarizes the features of the Cortex-R4F and some ARM, ARC, MIPS, and Tensilica processor cores available now.

Overall, ARM's Cortex-R4F is an incremental but important upgrade of the Cortex-R4. It paves the way for ARM to win more-demanding designs in automotive controllers and other embedded systems that need good floating-point performance, fault tolerance, and flexible core-configuration options.

The real challenge, however, will be displacing the processors already entrenched in automotives. Renewed cooperation between IBM and Freescale is strengthening the

Feature	ARM Cortex-R4F	ARM Cortex-R4	ARC ARC 625D	ARC ARC 750D	MIPS MIPS32 24Kf	Tensilica Xtensa 6
Architecture	ARMv7	ARMv7	ARCompact	ARCompact	MIPS32	Xtensa
Core Freq	400MHz**	400MHz**	210–340MHz**	330–460MHz**	500–660MHz**	350–400MHz
Pipeline Depth	8 stages	8 stages	5 stages	7 stages	8 stages	5 stages
Branch Predict	Dynamic	Dynamic	Static	Dynamic	Dynamic	—
Instr Length	32 bits	32 bits	32 bits	32 bits	32 bits	24 bits
Short Instr	16 bits (Thumb-2)	16 bits (Thumb-2)	16 bits	16 bits	16 bits	16 bits
DSP Instr	ARMv6 SIMD	ARMv6 SIMD	Yes	Yes	Optional (24KEf)	Yes
Custom Instr	—	—	Optional	Optional	Optional	Optional
FPU	Optional 32/64 bits IEEE 754	—	Optional 32 or 64 bits Partial IEEE 754	Optional 32 or 64 bits Partial IEEE 754	Optional 32/64 bits IEEE 754	Optional 32 bits No IEEE flags
MMU/MPU	Optional MPU	Optional MPU	Optional MPU	MMU	Optional MMU	Optional MMU
Caches	0K–64K	0K–64K	0–32K	8K–64K	0–64K	0–32K
TCM†	0–8MB	0–8MB	0–512K	0–512K	2 x 0–1MB	2 x 0–256K
Error Correction (Local Memory)	Optional Parity + ECC Caches + TCM	Optional Parity only Caches + TCM	—	—	Optional Parity only Caches, scratchpads	—
NMI‡	Yes	Yes	Yes	Yes	Yes	Yes
Bus Interface	AMBA-3 AXI	AMBA-3 AXI	AMBA AXI	AMBA AXI	OCP 2.x	Xtensa PIF AMBA AHB-Lite
Configurability	Medium	Medium	High	High	High	Highest
Size (Base)	0.86mm^2 90nm G**	0.8mm^2 90nm G**	0.35mm^2 90nm G**	0.75mm^2 90nm G**	0.77mm^2 90nm G**	20k gates
Power/MHz	0.25–0.33mW**	0.27–0.35mW**	0.06mW**	0.15mW**	0.43mW (w/caches)	0.04mW
Introduction	1Q07	2006	2005	2005	2004	2005

Table 3. Outside the automotive market, ARM's new Cortex-R4F will compete with 32-bit embedded-processor cores from ARC International, MIPS Technologies, and Tensilica. This table compares features of synthesizable processors that offer an FPU, an MPU or MMU, and similar performance characteristics. The ARC 625D and ARC 750D are preconfigured versions of the ARC 600 and ARC 700 cores, respectively. The Cortex-R4F is the only processor in this group with integrated parity checking and ECC. Purple text highlights key differences between the Cortex-R4F and earlier Cortex-R4. As always, take the area and power estimates with a large grain of salt. *Generic 0.13-micron CMOS process, worst case. **Generic 90nm CMOS process, worst case. †Tightly coupled memory, also called scratchpad memory. ‡Nonmaskable interrupts.

Power Architecture (formerly PowerPC), which has a solid position in automotives, particularly in the U.S. (See *MPR 8/21/06-01*, "The New Power Architecture.") In Europe, ARM's home territory, Infineon is a formidable competitor. Automobile manufacturers are famous for their lengthy design cycles and resistance to switching suppliers. Having the requisite features at the processor level is merely the first step in a marketing campaign that will likely take years to win major new designs. ◇