Proc. IEEE Power Driven Microarchitecture Workshop, pp. 145-150, Barcelona, Spain, 28th June 1998.

Designing the Low-Power M•CORE^{TM} Architecture

Jeff Scott, Lea Hwang Lee, John Arends, Bill Moyer

M•CORE Technology Center
Semiconductor Products Sector
Motorola, Inc. TX77/F51,
7600-C Capital of Texas Highway, Austin, TX 78731
{jscott,leahwang,arends,billm}@lakewood.sps.mot.com

Abstract

The M•CORE microRISC architecture has been developed to address the growing need for long battery life among today's portable applications. In this paper, we will present the low-power design techniques and architectural trade-offs made during the development of this processor. Specifically, we will discuss the initial benchmarking, the development of the Instruction Set Architecture (ISA), the custom datapath design, and the clocking methodology . Finally, we will discuss two system solutions utilizing the M•CORE processor, presenting power, area, and performance metrics.

1 Introduction

There is significant effort directed toward minimizing the power consumption of digital systems. Minimization techniques have been applied to various levels of abstraction from the circuit to the system level. The industry, coupled with contributions from the research community, has developed new architectures and microarchitectures [22,23], derivatives of existing architectures [1,15], compiler and compression optimizations [16,17], enhanced process technology [10], and new tools [11] and design techniques [5,7,14] to enhance overall system performance, which in this context translates to a decrease in milliwatts per MHz.

Existing architectures are not suitable for low-power applications due to their inefficiency in code density, memory bandwidth requirements, and architectural and implementation complexity. This drove the development of the new general purpose microRISC M•CORE architecture [18,21], which was designed from the ground up to achieve the lowest milliwatts per MHz. The M•CORE instruction set was optimized using benchmarks common to embedded applications coupled with benchmarks targeted specifically for portable applications. Compilers were developed in conjunction with the instruction set to maximize code density.

The benchmarks used to drive the development of the M•CORE architecture, as well as for making design trade-offs, are presented in Section 2. In Section 3, an overview of the instruction set is presented. Section 4 presents implementation details and various design methodologies. Section 5 presents power consumption statistics. Two system solutions based on the M•CORE microprocessor are presented in Section 6. Section 7 summarizes the paper.

2 Benchmarks

Embedded and portable benchmarks were used to make design trade-offs in the architecture and the compiler. The Powerstone benchmarks, which include paging, automobile control, signal processing, imaging, and fax applications, are detailed in Table 1.

Table 1: Powerstone benchmark suite

Benchmark	Instr. Count	Description
auto	17374	Automobile control applications
bilv	21363	Shift, and, or operations
blit	72416	Graphics application
compress	322101	A Unix utility
crc	22805	Cyclic redundancy check
des	510814	Data Encryption Standard
dhry	612713	Dhrystone
engine	986326	Engine control application
fir_int	629166	Integer FIR filter
g3fax	2918109	Group three fax decode (single level image decompression)
g721	231706	Adaptive differential PCM for voice compression
jpeg	9973639	JPEG 24-bit image decompression standard
pocsag	131159	POCSAG communication protocol for paging application
servo	41132	Hard disc drive servo control
summin	3463087	Handwriting recognition
ucbqsort	674165	U.C.B. Quick Sort
v42bis	8155159	Modem encoding/decoding
whet	3028736	Whetstone

3 Instruction Set Architecture

3.1 Overview

The M•CORE instruction set is designed to be an efficient target for high-level language compilers in terms of code density as well as execution cycle count. Integer data types of 8, 16 and 32-bits are supported to ease application migration from existing 8 and 16 bit microcontrollers. A standard set of arithmetic and logical instructions are provided, as well as instruction support for bit operations, byte extraction, data movement, and control flow modification.

A small set of conditionally executed instructions are available, which can be useful in eliminating short conditional branches. These instructions utilize the current setting of the processor's condition bit to determine whether they are executed. Conditional move, increment, decrement, and clear operations supplement the traditional conditional branch capabilities.

The M•CORE processor provides hardware support for certain operations which are not commonly available in low-cost microcontrollers. These include single-cycle logical shift (LSL, LSR), arithmetic shift (ASR), and rotate operations (ROTL), a single cycle find-first-one instruction (FF1), a hardware loop instruction (LOOPT), and instructions to speed up memory copy and initialization operations (LDQ,STQ). Also provided are instructions which generate an arbitrary power of 2 constant (BGENI, BGENR), as well as bit mask generation (BMASKI) for generating a bit string of '1's ranging from 1 to 32 bits in length. Absolute value (ABS) is available for assisting in magnitude comparisons.

Because of the importance of maximizing real-time control loop performance, hardware support for multiply and divide is also provided (MULT, DIVS, DIVU). These instructions provide an early-out execution model so that results are delivered in the minimum time possible. Certain algorithms also take advantage of the bit reverse instruction (BREV), which is efficiently implemented in the barrel shifter logic [3].

3.2 16-Bit vs. 32-Bit

System cost and power consumption are strongly affected by the memory requirements of the application set. To address this, the M•CORE architecture adopts a compact 16-bit fixed length instruction format, and a 32-bit Load/Store RISC architecture. The result is high code density which reduces the total memory footprint, as well as minimizing the instruction fetch traffic.

Benchmark results on a variety of application tasks indicate that the code density of the M•CORE microRISC engine is higher than many CISC (complex instruction set computer) designs, in spite of the fixed length nature of the encodings. Fixed-length instructions also serve to reduce the CPU's control unit complexity, since instructions have a small number of well structured formats, thus simplifying the decoding process.

3.3 Low-Power Instructions

The M•CORE processor minimizes power dissipation by utilizing dynamic power management. DOZE, WAIT, and STOP power conservation modes provide for comprehensive system power management. These modes are invoked via issuing the appropriate M•CORE instruction. System level design dictates the operation of various components in each of the low power modes, allowing a flexible set of operating conditions which can be tailored to the needs of a particular application. For example, a system may disable all unnecessary peripherals in DOZE mode, while in STOP mode, all activity may be disabled. For short term idle conditions, the WAIT mode may disable only the CPU, leaving peripheral functions actively operating [18].

3.4 Instruction Usage

The M•CORE ISA was profiled by running the Powerstone benchmark suite on a cycle accurate C++ simulator. Table 2 shows the percentage of dynamic instructions utilizing the adder and barrel shifter, as well as the percentage of change of flow and load/store instructions.

Table 2: Dynamic instruction percentages

Type	Dyn. Inst. Percentage
Adder usage	50.23%
Barrel shifter usage	9.68%
Change of flow instructions[a]	17.04%
Load/store instructions	22.46%

a. 83.5% of change of flow instructions are taken.

4 Processor Implementation

The initial implementation of the M•CORE architecture was targeted at sub-2.0 volt operation from 0-40 MHz and achieves both low-power consumption and high-performance in a 0.36-micron low-leakage static CMOS process. In addition, a die size of just over 2 mm^2 was achieved by utilizing a synthesizable control section and full-custom datapath. The first implementation is also being produced in a 0.31-micron process for a cellular application and has been targeted for production in a 0.25-micron process for other embedded applications. Other low-power process technologies are being investigated including a Silicon On Insulator (SOI) process [2].

Power consumption was minimized in the first implementation of the M•CORE architecture by: (i) gating clocks; (ii) utilizing delayed clocks to minimize glitching; and (iii) optimizing the clock tree. Powermill [12], Spice, and Focus [8], a transistor sizing power optimization tool, were used to optimize the circuits.

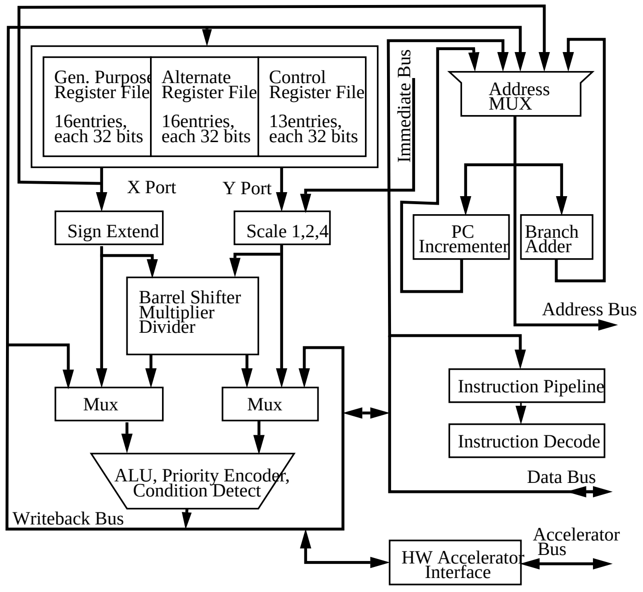

Figure 1: M•CORE datapath

4.1 Datapath Design

The M•CORE datapath consists of a register file, an instruction register, an adder, a logical unit, a program counter, a branch adder, and a barrel shifter. A block diagram of the datapath is shown in Figure 1.

4.1.1 Synthesizable vs. Custom

Synthesis methodologies have the advantage of decreased time to market, fewer resource requirements, and the ability to quickly migrate to different technologies. Custom designs lack those benefits, but can achieve faster performance, consume less power, and occupy less area.

Research was conducted to determine whether to pursue a synthesizable or custom datapath design. An energy-delay metric [13] was used to develop a custom 32-bit adder. Our studies showed that in going from a custom to a synthesized adder, transistor count increased by 60%, area increased by 175%, and power consumption increased by 40%. This research coupled with similar block-level comparisons led to the development of a full-custom datapath.

4.1.2 Speed, Area and Power Trade-off

Focus [8] was used to optimize speed, area and power in the custom datapath. Focus takes a user-specified circuit and timing constraints and generates a family of solutions which form a performance/area trade-off curve. Given a set of input vectors, Focus allows the user to invoke Powermill interactively to measure power at any given solution. The user is then allowed to select the solution with the desired performance, area, and power characteristics.

Focus uses a dc-connected component-based static timing analysis technique to propagate the arrival times in a forward pass through the circuit, and the required times in a backward pass. At each node, a slack is generated as the difference between the required and the arrival times (negative slack denotes a violation of a constraint).

Focus performs a hill climbing search starting from a minimum sized circuit (or a given initial circuit). In each step, Focus picks a set of candidate transistors to size based on their sensitivity to the overall path delay, and iteratively and gradually approaches a solution that will meet the timing constraints[8].

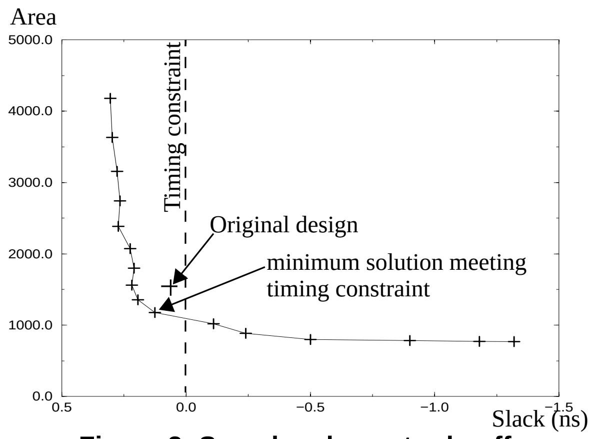

Figure 2: Speed and area trade-off

Figure 2 shows an example of how Focus was used to design the logical bit in the ALU. The vertical axis is expressed in some normalized area cost, while the horizontal axis represents the slack in nanoseconds.

The figure shows that Focus gives a family of solutions. It also shows that faster timing can be achieved at the expense of higher area cost. Utilizing this family of solutions, a solution was selected to meet the timing constraints with optimal area and power savings.

4.1.3 Floorplanning

Floorplanning of the datapath blocks was based on bus switching activity, timing, and block usage percentages derived from the Powerstone benchmarks. Blocks having output busses with high switching activity, tight timing requirements, and high usage were placed close together to minimize the routing capacitance. When searching through the solution space of block placement, timing constraints were used as hard constraints, while switching activities, weighted by their appropriate routing distances and block usage, were used as a cost function in the iterative search process.

In the final floorplan, shown in Figure 3, each horizontal line represents a multiple bit (up to 32 bits) data bus route. The datapath could be roughly partitioned into two halves, separated by the register file: the left half includes the address and data interface, the instruction register, the branch adder, and the address generation unit; the right half includes the execution units.

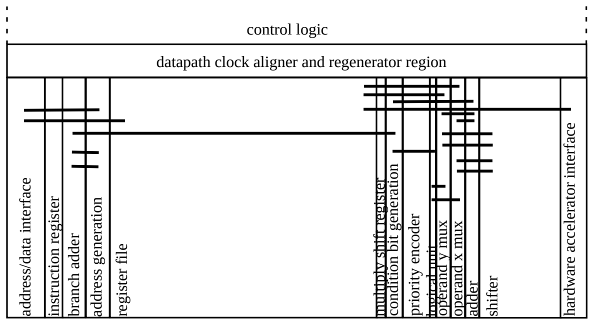

Figure 3: Datapath floorplan

4.1.4 Register File Design

An architectural decision was made to have sixteen general purpose registers based on the limited opcode space. An alternate set of sixteen 32-bit registers is provided as well, to reduce the overhead associated with context switching and saving/restoring for time-critical tasks. This alternate register file can be selected via a bit (AF) in the processor status register (PSR). Upon detecting an exception, the hardware places bit 0 of the exception vector table entry into the AF bit in the PSR. This allows the exception handler to select either register file for exception processing with no cycle penalty [20]. Hardware does not preclude access to the alternate register file in user mode, resulting in a larger working set of registers. This can result in a reduced number of memory fetches, thereby saving power.

In supervisor mode, an additional 5 supervisor scratch registers can be accessed by means of the move-to-control-register (MTCR) and move-from-control-register (MFCR) instructions.

The register file also includes: (i) feed forwarding logic; (ii) immediate muxes to produce various constants; and (iii) scale logic for computing load/store addresses.

4.1.5 Barrel Shifter Design

As noted earlier, approximately 10% of the instructions executed in the Powerstone benchmarks utilize the barrel shifter. In addition, barrel shifters are typically one of the highest power consuming modules in datapath designs. A comparison was made between two competing barrel shifters to determine which one offered the least current drain within a given performance requirement.

"Snaked Data" Barrel Shifter

"Snaked data" barrel shifters contain a 32x32 array of transistors. For precharged shifter outputs, the array uses nmos transistors. The term "snaked data" comes from the fact that input data (IL[2:0],IR[3:0]) is driven diagonally across the array, as shown in Figure 4. Control signals (S[3-0]) to select the amount of the shift are driven vertically and outputs (O[3:0]) are driven horizontally. Data enters the shifter through multiplexers with selection based on the type of shift operation, either rotates, logicals, or arithmetics. The control signals are generated from a 5-32 decoder for 32-bit shifters, where 5 bits represent the amount to shift and can be inverted to select shifts in the opposite direction.

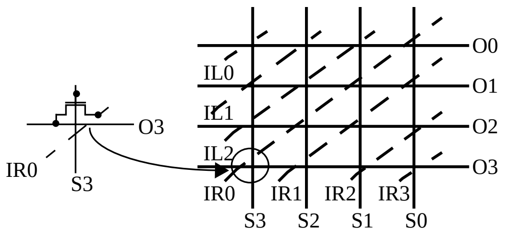

Figure 4: 4-bit "snaked data" barrel shifter

"Snaked Control" Barrel Shifter

"Snaked control" barrel shifters contain a 32x32 array of transistors. For precharged shifter outputs, the array uses nmos transistors. The term "snaked control" comes from the fact that control signals (SL[3:1],SR[3:0]) are driven diagonally across the array as shown in Figure 5. Shifter outputs (O[3:0]) are driven vertically and inputs (I[3:0]) are driven horizontally. Control signal assertion is based on the type of shift operation, either rotates, logicals, or arithmetics and the amount of the shift. The control signals require a 5-32 decoder, where 5 bits represent the amount of shift, as well as a greater-than decoder for arithmetic operations.

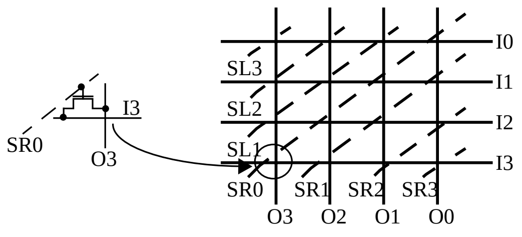

Figure 5: 4-bit "snaked control" barrel shifter

Barrel Shifter Selection

There are three components of routing in either shifter design. Table 3 shows the routing breakdown for the two 32-bit shifters.

Table 3: Snaked control vs. snaked data routing

Shifter Type	Input Data Routing	Output Data Routing	Control Routing
Snaked control	12,896	173,056	171,962
Snaked data	171,962	24,576	173,056

Control signal power consumption is negligible in both cases, since the selections are one-hot. Even though the total length of power consuming data routing for the two designs is nearly the same, the "snaked control" barrel shifter actually has significantly less routing capacitance, 5.9 pF vs. 7.4 pF, because of metal layer usage.

It would appear that the "snaked control" would be the ideal choice for a power conscious design. However, further analysis of the Powerstone benchmarks was required. Many benchmarks perform similar types of shifts in succession. In fact, 64% of the shifts are either logical shift left to logical shift left, or logical shift right to logical shift right. Under these conditions, the "snaked data" shifter maintains constant data input into half of the array throughout the entire algorithm. In these circumstances, the "snaked data" shifter toggles over 30% less capacitance then the "snaked control" barrel shifter.

4.2 Clocking System Design

Clock gating saves significant power by eliminating unnecessary transitions of the latching clocks. Furthermore, it also eliminates unnecessary transitioning of downstream logic connected to the output of the latches. In the datapath, delayed clocks are generated for late arriving latch input data in order to further minimize downstream glitching.

Power due to clocking can vary anywhere from 30% to 50% in a power-optimized design [4]. As more and more datapath power optimizations were added to the design, the percentage of power due to clocking increased. Therefore, it was crucial to aggressively attack clock power consumption.

The M•CORE processor uses a single global clock with local generation of dual phase non-overlapping clocks. The global clock proceeds through two levels of buffering/gating; a clock *aligner* and *regenerator*, as shown in Figure 6 and Figure 7 respectively. The aligner generates the dual phase clocks, while the regenerator provides local buffering.

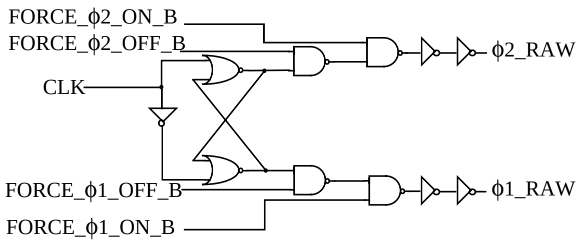

Figure 6: Clock aligner

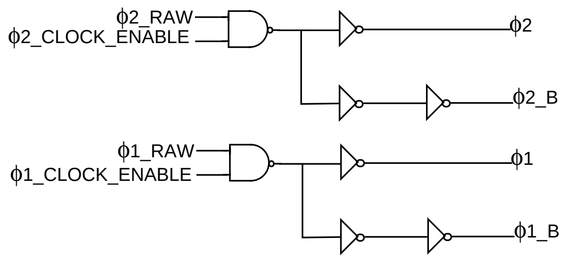

Figure 7: Clock regenerator

Clock gating can be performed at the aligners and/or the regenerators. This allows for complete or partial clock tree disabling. Besides shutting down the system clock input completely, gating at the aligners allows for maximum power savings by shutting down a collection of regenerators. In the instruction decoder, for example, the clocking for over 60 latches and flip-flops is disabled during pipeline stalls.

5 Power Measurement

Powermill and Pathmill were used to perform the following tasks: (i) measuring power usage of each block to detect any design anomalies; (ii) detecting nodes with excessive DC leakage current; (iii) locating hot spots in the design; and (iv) identifying nodes with excessive rise and fall times. Various design changes were made based on these findings.

5.1 Processor Power Distribution

M•CORE processor power was measured using Powermill on a back-annotated taped out implementation of the architecture. Clock power is 36% of the total processor power consumption, as shown in Figure 8. It is important to note that due to measurement constraints, the datapath regenerator clocking power is included in the datapath percentage, not the clocking power percentage. This clocking power would push the clock power percentage even higher. This is a good indication that the datapath and control sections are well optimized for power. It is also important to note that through utilization of the low-power mode instructions, the system clock can be completely shut down.

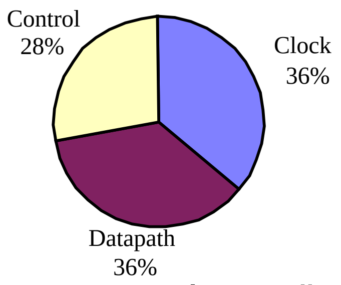

Figure 8: Processor total power distribution

5.2 Datapath Power Consumption

Figure 9 shows the breakdown of the datapath power consumption. The register file consumes 16% of total processor power consumption and 42% of the datapath power consumption. It is the largest consumer of datapath power and occupies 46% of the datapath area as well. The barrel shifter consumes less than 4% of the total processor power consumption and 8% of the datapath power consumption.

The address/data interface contains the local bus drivers. The instruction register drives the instruction decoder in the control section. The operand muxes drive the ALU, priority encoder, and condition detect logic. The address generation unit contains the program counter and dedicated branch adder circuitry (see Figure 1).

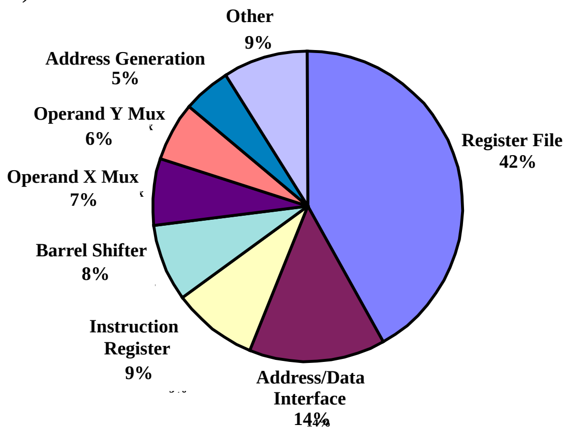

Figure 9: Datapath power consumption

6 System Solutions

6.1 MMC2001

The MMC2001 low power platform incorporates an M•CORE integer processor, an on-chip memory system, a OnCE^{TM} debug module, an external interface module, and peripheral devices [19]. It includes a section of 20K gates that are customizable. The MMC2001's production version is implemented in a 0.36 micron, triple-layer metal low-leakage static CMOS process. The on-chip memory system consists of 256 KBytes of ROM and 32 KBytes of SRAM. The device measures 7.23 mm on a side or 52 sq. mm. The part is packaged in a 144 pin Thin Quad Flat Package (TQFP) and achieves 34 MHz performance at 1.8 V.

The MMC2001, when running out of internal SRAM, consumes on average less than 30 mA at 2.0 V, which translates into less than 60 mW at 34 MHz for the complete system. On average, this implementation of the M•CORE processor consumes 7 mA, which translates to a 0.40 mW/MHz rating. When the part is put into STOP mode, the part consumes less than 50 microamps with the clock input shut down. When the part goes into a low voltage detect condition with just the oscillator, SRAM backup and time-of-day timer running, the part consumes less than 5 microamps.

6.2 MMC56652

The MMC56652 Dual-Core Baseband Processor incorporates an M•CORE integer processor, an on-chip memory system, a OnCE^{TM} debug module, an external interface module, and peripheral devices [9]. It includes a 56600 Onyx Ultra-Lite 16-bit digital signal processor, program and data memory, and peripheral devices. The MMC56652's production version is implemented in a 0.31 micron, triple-layer metal static CMOS process. This device consists of 8 KBytes of ROM and 2 KBytes of SRAM to support the M•CORE processor. The chip measures 7.4 mm on a side or 55 sq. mm. The part is packaged in a 196 plastic ball grid array (PBGA) and was designed for 16.8 MHz performance at 1.8 V.

The MMC56652 when running out of internal SRAM consumes on average less than 9 mA at 1.8 V, which translates into less than 16.2 mW at 16.8 MHz for the complete system. On average, this implementation of the M•CORE processor consumes 2.8 mA, which translates to a 0.30 mW/MHz rating. The part consumes less than 60 microamps in STOP mode.

7 Conclusion

The M•CORE architecture development represents four years of research in the area of low-power embedded microprocessor and system design. This research encompassed all levels of abstraction from application software, system solutions, compiler technologies, architecture, circuits, and process optimizations.

The instruction set architecture was developed in conjunction with benchmark analysis and compiler optimization. A full-custom datapath design with clock gating was utilized to minimize the power consumption. Two system solutions integrating the M•CORE microprocessor were presented along with power, area, and performance metrics.

Further research will involve allocating unutilized opcode space, along with additional compiler enhancements. Benchmarks will continue to be added to the Powerstone suite reflecting future portable low-power applications. Efforts will be made to drive novel design techniques and process technologies into a production environment in order to achieve higher levels of power reduction and performance.

8 References

[1] *An Introduction to Thumb*, Advanced RISC Machines Ltd., March 1995.

[2] D. Antoniadis, "SOI CMOS as a Mainstream Low-Power Technology," *Proc. IEEE Int'l. Symp. Low Power Electronics and Design*, August 1997, pp. 295-300.

[3] J. Arends, J. Scott, "*Low Power Consumption Circuit and Method of Operation for Implementing Shifts and Bit Reversals*," U. S. Patent, number 5,682,340, October 28, 1997.

[4] R.S. Bajwa, N. Schumann, H. Kojima, "Power Analysis of a 32-bit RISC Microcontroller Integrated with a 16-Bit DSP," *Proc. IEEE Int'l Symp. Low Power Electronics and Design*, August, 1997, pp. 137-142.

[5] L. Benini, G. Micheli, "Transformation and Synthesis of FSMs for Low-Power Gated-Clock Implementation," *Proc. IEEE Int'l. Symp. on Low Power Design*, 1995, pp. 21-26.

[6] J. Bunda, D Fussell, R. Jenevein, W. C. Athas, "16-Bit vs. 32-Bit Instructions for Pipelined Microprocessors," University of Texas, Austin, October 1992.

[7] A. Chandrakasan, S. Sheng, R. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, April 1992, pp. 473-484.

[8] A. Dharchoudhury, D. Blaauw, J. Norton, S. Pullela, J. Dunning, "Transistor-level Sizing and Timing Verification of Domino Circuits in the PowerPC Microprocessor," *Proc. IEEE Int'l. Conf. Computer Design*, Austin, Texas, October 1997.

[9] *DSP 56652 User's Manual*, Motorola Inc., 1998.

[10] D. Frank, P. Solomon, S. Reynolds, and J. Shin, "Supply and Threshold Voltage Optimizations for Low Power Design," *Proc. IEEE Int'l. Symp. Low Power Electronics and Design*, August 1997, pp. 317-322.

[11] J. Frenkil, "Issues and Directions in Low Power Design Tools: An Industrial Perspective," *Proc. IEEE Int'l Symp. Low Power Electronics and Design*. August 1997, pp. 152-157.

[12] C. Huang, B. Zhang, A. Deng, B. Swirski, "The Design and Implementation of Powermill," *Proc. IEEE Int'l Symp. Low Power Design*, 1995, pp. 105-109.

[13] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-Power Digital Design," *Proc. IEEE Int'l. Symp. on Low Power Electronics"*, October, 1994, pp. 8-11.

[14] M. Igarashi, K. Usami, K. Nogami, F. Minami, "A Low-Power Design Method Using Multiple Supply Voltages," *Proc. IEEE Int'l. Symp. Low Power Electronics and Design*, August 1997, pp. 36-41.

[15] K. Kissell, "MIPS16: High Density MIPS for the Embedded Market," Silicon Graphics MIPS Group, 1997.

[16] M. Kozuch and A. Wolfe, "Compression of embedded system programs," *Proc. IEEE Int'l. Conf. on Computer Design*, 1994.

[17] C. Lefurgy, P. Bird, I. Chen, and T. Mudge, "Improving Code Density Using Compression Techniques," *Proc. IEEE Int'l. Symp. on Microarchitecture*, December, 1997, pp. 194-203.

[18] *M•CORE Reference Manual*, Motorola Inc., 1997.

[19] *MMC2001 Reference Manual*, Motorola Inc., 1998.

[20] W. Moyer, J. Arends, patent pending, "*Method and Apparatus for Selecting a Register File in a Data Processing System*," U. S. Patent, filed September, 1996.

[21] W. Moyer, J. Arends, "RISC Gets Small," *Byte Magazine*, February 1998.

[22] "NEC Introduces 32-bit RISC V832 Microcontroller," NEC Inc., May 1998.

[23] J. Slager, "SH-4: A Low-Power RISC for Multimedia and Personal Access Systems," *Microprocessor Forum*, October 1996.

M•CORE is a trademark of Motorola, Inc.
OnCE is a trademark of Motorola, Inc.