Low Power Methodology and Design Techniques for Processor Design

J. Patrick Brennan, Alvar Dean, Stephen Kenyon, and Sebastian Ventrone
IBM Microelectronics, 1000 River Street 862C, Essex Junction VT 05452
brennanp@us.ibm.com

ABSTRACT

IBM's ASIC design methodologies is used to develop a low power microprocessor for the mobile (battery powered) marketplace. The design called for a reduction of active power by a factor of 10 times from an estimate of a product designed in a standard 3 volt ASIC design system. An overview of the design methodology and some of the innovative power reduction techniques are presented.

INTRODUCTION

As the portable, battery-operated, electronics market moves to computational-intensive products like cellular telephones and notebook computers, the need to focus on low-power design becomes critical to extend battery life. The small feature sizes and lower operating voltages of today's advanced semiconductor processes inherently reduce power consumption. The small feature sizes reduce circuit capacitance (wire lengths and parasitics), which directly reduces dynamic (switching) power consumption, while the lower voltages reduce power by the square of the voltage reduction. Combining these technology features with low-power design techniques and ASIC design systems is key to introducing "Core+ASIC" low-power designs for the mobile market.

ISLPED98, Monterey, CA, USA

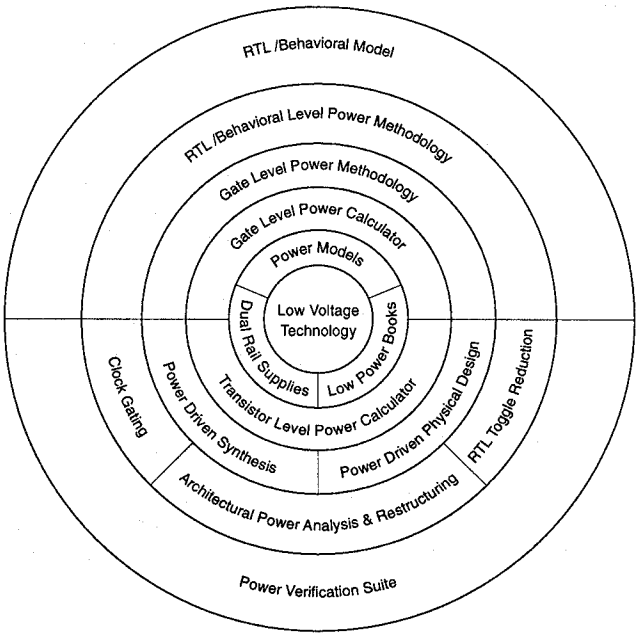

Figure 1. The Power Wheel

The power wheel depicts all major components needed for low-power design. At the center are the low-power technology and supporting books models. As you move up in levels of abstractions, you go into the outer rings from transistor to gate to register transfer (RT) levels.

Low-power product design has traditionally been a manual process aided by some Electronic Design Automation (EDA) software tools. Designing for the lowest possible power consumption requires addressing all phases of the electronic design process: architectural definition, logic entry, floor planning, circuit design, and optimal use of low-power process technologies. An accurate power estimator, coupled with useful power metric reporting is key to delivering first-pass hardware that meets the targeted power specification.

Typically, an ASIC cell-based design methodology was not considered an option in the design of complex integrated products with low-power characteristics. In this article, we present the process and techniques of IBM's Blue Logic ASIC design

methodologies to develop a low-power microprocessor for the mobile (battery powered) marketplace.

PROJECT OVERVIEW

The design of this microprocessor was done in two phases, First the functionality and performance targets were achieved, and second, a power reduction effort was undertaken to meet the power consumption objectives. The power consumption target called for a reduction of active power by a factor of 10 from an estimate of the power of a product designed in a standard 3 V ASIC design system.

To achieve the power reduction target, six major efforts were undertaken:

1. Utilize a IBM ASIC design system that supported dual rail power supplies, (core to run at a lower voltage than the I/O circuits).
2. Create method to enable fast and effective power consumption feedback at both the Register Transfer (RT) and gate levels.
3. Change design architecture and structure to reduce dynamic power.
4. Employ new and innovative methods for active power reduction, including pseudo-microcode, a new logic structure for controlling data paths, and innovative logic analysis combined with a very wide word (VWW) control system.
5. New low power IBM core library elements (RAMs, ROMs, Phase Locked Loop (PLL), voltage regulator, and one new logic book type).
6. The physical design team made extensive use of bit-stacking and latch/clock splitter "power groups" .

Power and performance are predicted correctly by IBM's ASIC design methodology tools. The resulting product is fully LSSD testable..

Using the techniques described in this paper, our team was able to meet the power consumption target for the microprocessor. The table below shows the percent of overall power savings observed for our design with the low-power design techniques described in this paper, excluding power savings achieved by changes to custom macros, such as the ROM, PLL, and the RAM, which were very significant. Actual power savings may vary from design to design.

Technique	Saving
Low Power Synthesis	15%
Clock Gating	8%
Logic/Architectural Changes	45%
Voltage Reduction	32%

POWER REDUCTION METHODOLOGY

A power reduction methodology (Figure 2) must support power consumption feedback at the HDL, gate, or transistor level. Varying degrees of power calculation accuracy are expected at the various levels of design abstraction. Basic correlation must exist to assure that the benefits of power reduction changes at the HDL level result in corresponding reductions at the gate level and transistor level.

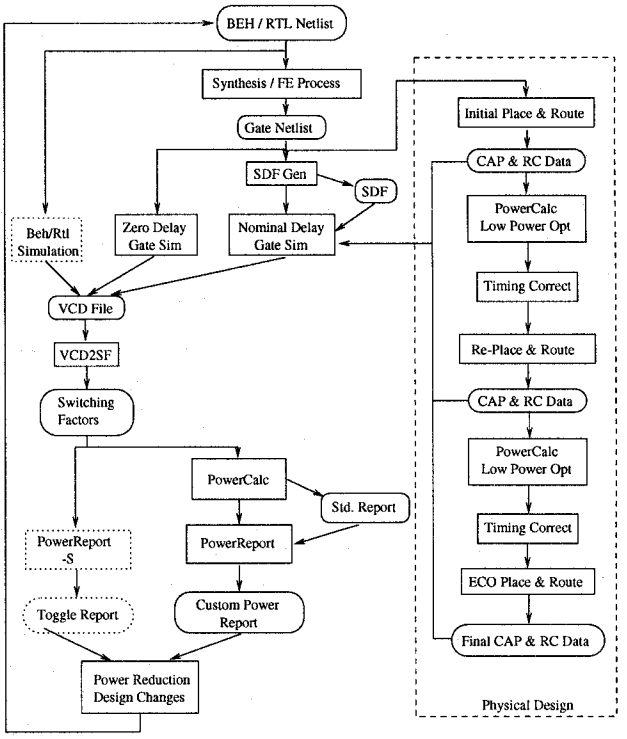

Figure 2. Low Power Design Methodology

The design methodology allows for early power consumption feedback at the BEH/RT level in the form of node toggle reports and power calculations at the gate level with either zero, unit, or nominal delay simulation results. Physical design parasitics can be used at the gate level for improved power estimation.

Accuracy at the gate level and good correlation to hardware are key factors for a reliable power reduction methodology. A power calculator's results are only as good as the available switching factor information. The power estimating tools used by the design team report the power information in a form (toggles) so that the team could make the most informed decisions about where to prioritize their power reduction efforts.

Other factors that substantially effect the power calculation results are: the data values used in data-path or arithmetic units, and the values and frequency of primary input and output switches. Ideally, the logic synthesis process must be power driven (in addition to area and timing driven) to observe a power budget.

Automatic power optimization algorithms that can be applied at the gate level are very desirable.

Power Verification Suite: Finding a set of representative applications that will yield power consumption close to the average power of the hardware is difficult. In battery-powered embedded applications, the microprocessor function is limited to the application code in the ROM. The range of applications is typically bounded; embedded applications spend most of their powered-up cycles in certain segments of code. This bounded code can confidently be used to characterize average dynamic operating power and identify instructions with high usage frequency.

Behavioral/RT Power Estimation/Reduction: Since dynamic power consumption is directly proportional to the amount of switching activity, at the behavioral (BEH) or RT levels, switching activity can approximate the power consumption. Net and bus switching factors obtained from BEH/RT level logic simulation can identify hot spots on which the designer's efforts should be focused. Once a baseline behavioral switching factor information is obtained, pass-to-pass comparisons of the BEH/RT level changes can verify the reduction of switching activity and thus dynamic power.

Gate Level Power Calculation and Optimization: At the gate level, an accurate power calculation is possible, limited only by the accuracy of the gate power models and the extracted parasitics. Early on in the design process, estimated parasitics coupled with a zero-delay simulator provide enough accuracy, while post-wiring parasitics and nominal delay simulation are required for more accurate calculations later on. Logic synthesis selects logic gate size based on net capacitance and/or slew driven power minimization.

Once estimated parasitcs are obtained from a place and route of the gate level netlist, a nominal delay simulation is done to extract the network switching factors from the Power verification suite. The slew driven power optimization tool is run to find the optimal slew rate and gate sizes for power consumption that may result in higher or lower powered books being selected (gates powered up or down). At this point, we have obtained minimum crossover current during CMOS switching while meeting the network timing constraints.

The correlation between BEH/RT level power estimates via node toggles and gate level power calculations varied from design unit to design unit from 5% to over 30% difference.

POWER CONSUMPTION REPORTING

The power consumption report is a key component of the power reduction process. A power consumption report must be detailed and formatted such that a designer can promptly implement power reduction changes. Too low a level of detail, such as at the cell level, may be useless to a designer who is thinking in terms of high-level blocks. We developed a customizable power reporting program that abstracts the detailed power information from the PowerCalc report and presents it at a level delineated by the designers. Power can be reported not only at the net or gate level, but also at the bus, data-path, unit, or sub-unit abstraction levels, all under user control.

Since the clock distribution network can consume anywhere from one quarter to one half of the active power in a design, special attention must be given to the clock distribution network. Techniques such as clock gating have traditionally been used to manage clock distribution power. Our ASIC clock distribution methodology calls for a single clock to be distributed through a re-driven network to clock splitters, which create two non-overlapping, out-of-phase clocks. These clocks are then used to drive master/slave latches. We developed a specialized clock distribution power reporting program that post-processes the PowerCalc power report and gathers all the clock network power by unit and breaks it down by driver and splitter power. This data is very helpful when prioritizing and structuring the clock gating logic and analyzing the clock distribution network.

DYNAMIC POWER REDUCTION

Cycle-to-Cycle Minimization: The two main approaches to RT level power reduction are first, to minimize the power required to implement a function, and second, to minimize how often the function needs to be executed. While the former approach reduces power, the latter can result in much larger power reduction. A new method of reducing AC power was devised to minimize cycle-to-cycle unnecessary toggles. In this method, next cycle control signals are derived as a function of next cycle function, along with the previous cycle state. By minimizing cycle-to-cycle functional toggle requirements, power can be saved independent of function implementation. The idea is a simple one, if a functional path is not required on the current cycle, ensure that all the control paths and data paths remain at the previous cycle's state, eliminating all node toggles.

Pseudo-Microcode: The IBM microprocessor design chose a decode method that differed from other microprocessors. The two common decoding techniques are distributed decode, where each logic control signal is derived as an independent cone of logic, and microcode, where each opcode is translated to an entry point address to a Read Only Memory (ROM) or Programmable Logic Array (PLA) that provides the complete set of control signals. We combined the simplicity of microcode with the flexibility of distributed decode into a style we call *pseudo-microcode.* For every opcode, all control signals that can be set by that opcode are bundled together as a total identity. These include the pipeline control, the address generation control, and the execute control bits. Upon each cycle boundary, a first stage very wide word (VWW) dispatch (Figure 4) is sent to n microcode units (22 for IBM microprocessor). The microcodes are decoded in each microcode unit for the instruction type. Once the instruction type is determined, then the value is set for every control signal within that microcode unit.

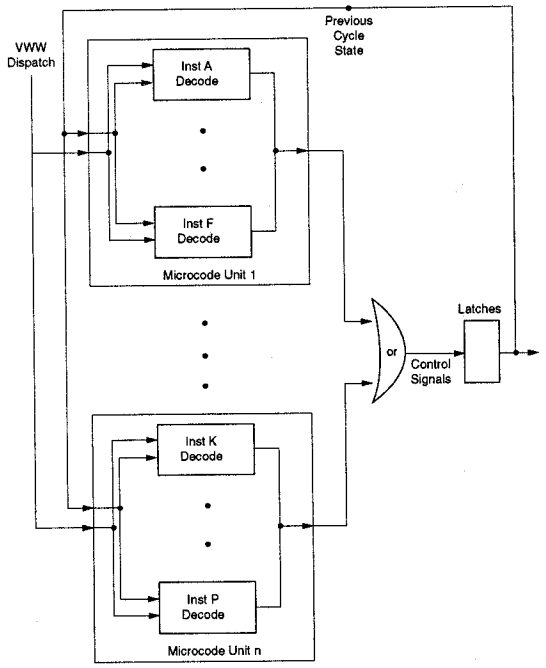

Figure 3. Pseudo-Microcode Structure

By combining current cycle decode (VWW) with previous cycle state each opcode becomes a function of the previous opcode. Only conflicting signals need to be resolved enabling the minimum amount of logic needed to execute an opcode.

For every instruction decode, the control signals can be set to 0, 1 value, or the previous value. By only changing the signals that are required for that function, the power of the machine can be reduced. Now each opcode becomes a function of the previous opcode (Figure 3). Only conflicting control signals need to be resolved. In the example below, during the first NOP instruction, the controls to the Arithmetic Logic Unit (ALU) are left in the ADD state even though the ALU is not needed to perform the NOP operation. The ALU controls are left in the SUB state during the NOP and MULT instructions following the SUB instruction.

EX: ADD/SUB (control signals are: Set ALU_Control ON, Set Register_Enable ON, Set MULT_Control ON)
NOP (looks like ADD, but Register_Enable OFF)
SUB (looks like ADD, but different ALU_Control)
NOP (looks like SUB, but Register_Enable OFF)
MULT (looks like SUB, but Register_Enable ON, MULT_Control ON)

etc.

With this method, the control signals for each unit remain in the same state and only change when a new state is required with the next opcode being executed essentially turning the processor into a giant state machine.

An advantage to this type of microcode is the ability to embed logic within the microcode, such as conditional checks within an opcode type. This can reduce the number of microcode addresses which may reduce the overall size of the decode unit, saving power.

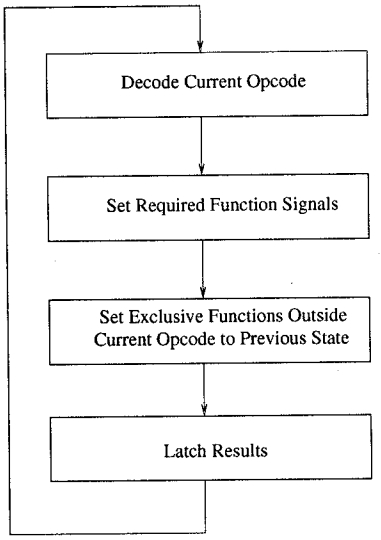

Figure 4. Pseudo-Microcode State Machine Decoding

A VWW feeding a *pseudo-microcode* function allows for state change minimization, embedded logic, passing of variables, and improved readability in a design of lower power and more flexible design structure.

CONTROLLING DATA PATHS

Clock Gating by Opcode Typing: Since the clock distribution network typically consumes a large percentage of the processor power, clock gating, where the clocks are turned off to portions of the network that do not require it, can save a lot of active power. This mechanism works well for data-flow logic, where clocking requirements can be predetermined at least one cycle ahead. Clock gating is difficult within the current cycle for control registers due to the random nature of the control logic. In addition, the clock gating signals must be valid halfway into the cycle to gate off the capture clock.

An automated method for generating clock gating signals the maximize the number of laches that are turned off every cycle was developed. This method, called opcode typing, searches the microcode function and groups control signals into clock gating groups with the goal of maximizing the number of opcodes for which the signals in the clock gating group stay constant. Type generation is done with a program that iteratively picks different groupings of signals to be typed together, analyses opcode coverage, and picks the type groupings that maximize opcode coverage with the minimum number of groupings.

Once optimum clock gating groups are defined and opcodes have been assigned to types, typed clock gating can begin. Early in the cycle, opcodes are pre-decoded to generate a type field corresponding to the groups of registers. The current opcode's type field is then compared to the previous cycle type field. If the previous cycle type is ON, and the current type is ON, then clock gating can be enabled since the control signals will be the same value from the previous opcode to the current opcode for that type. In the microprocessor decode, about 10 clock gating signals were generated for the registers, but this is entirely a function of

the number registers and the amount of type coverage that one is trying to obtain.

While this method of typing control signals by opcode/microcode/function cannot clock gate on a per-bit basis, a significant ratio of cycle-to-cycle clock gating can be obtained for control logic registers.

Data-Path Toggle Reduction: A common element in data-path processors is the bus multiplexer. The multiplexer selects one of several data buses to be driven onto a single output bus. If care is not taken, unwanted transitions on multiplexer inputs can propagate to the output buses and to downstream logic, even when the multiplexer is not used during the current cycle. The standard approach to multiplexer node toggle reduction is to hold previous value, or to force the multiplexer to a constant state when not in use. On our application, this approach did not yield any significant power reduction when applied to heavily used multiplexers. While power is saved when the multiplexer is not used for many cycles, if the multiplexer is required to change values frequently, no power savings are observed.

An alternative method was created for data-path multiplexing which guarantees that the multiplexer outputs will switch once and only once during the cycle when they are needed and never on cycles where the multiplexer is not functionally needed. In a normal data-path design, due to individual path delay differences, different bits of a bus will arrive at a different time within the cycle, which may result in repeated arithmetic or logical calculations downstream as an input bus settles on the final value for the cycle. To minimize data-path power, inputs to arithmetic units, such as multipliers, shifters and adders, should be applied only once during the cycle on which they are used.

To achieve these goals, a new *transition-once* multiplexer was designed (Figure 5). The transition-once multiplexer ensures a single transition on the output bus per cycle by guaranteeing that data is propagated through the multiplexer only when the input data buses are valid and stable. The transition-once multiplexer holds the previous value when the multiplexer is not selected or during transitions to a new value. If the multiplexer is not selected in the current cycle, the output is held at the previous cycle's state.

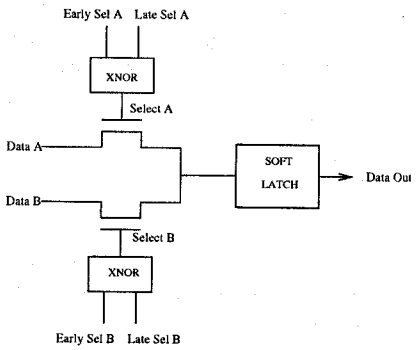

Figure 5. Two Port Transition-Once Multiplexer Schematic

Four rules exists for the transition-once multiplexer:

1. No new data until select off
2. No new select until new data
3. If no select, hold previous value
4. If same select, deselect (and hold previous value) during data transition time

To accomplish these four rules, two selects are defined for every data port. One select is called the fast select, and the second select is the slow select. The fast select was timed to be faster then the fastest data bit for that input port. The slow select was timed to arrive after the slowest bit of the same data port. The two selects are fed into an exclusive NOR (XNOR). Each select must be the same value (either both 0 or both 1) for the multiplexer input port to be gated to the output stage. Between the input stage and output stage is a soft latch, which holds the output state during either a selection to a new data port or when no pair of selects are active. If the same input port is to be selected from one cycle to another, then both selects are toggled (if both 0, then on the next cycle both would be 1). By toggling both selects, the input port is deselected during the input data transition. The fast select will go to the new state before the slow select, temporarily disconnecting the input port from the output during the transition of the input port. Once the slow select changes to the same value as the fast select, the data transition is complete, and the new value of the selected input port is allowed to pass to the output stage.

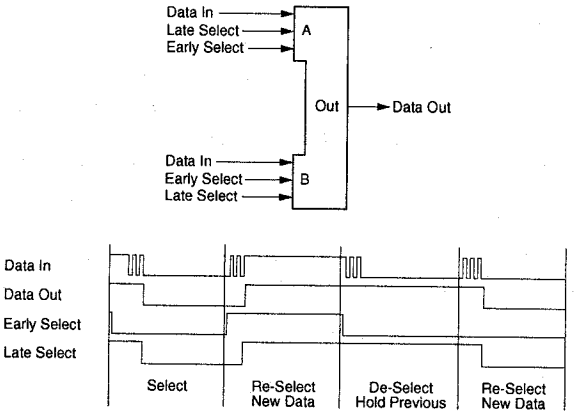

Figure 6. Two Port Transition-Once Multiplexer Timing Diagram

The transition-once multiplexer reduces toggles by filtering out transient changes in the input data from the output date. Figure 6 shows how the multiplexer is used to either select a data path, re-select the same data path on another cycle, or de-select a path to hold previous data. The late select must be timed to arrive no earlier then the fact input data transition.

By cutting down the intermediate transitions of data multiplexers and preventing false calculations of downstream logic from occurring, the transition-once multiplexer played a major role in the dynamic power reduction effort. Note that if the fast and slow selects are not timed to arrive at their optimal times, the only penalty will be a few toggles that pass thru the multiplexer.

A variation on the transition-once multiplexer was to use a transition-once buffer within random logic (Figure 7). At key pinch points in the design, a transition-once buffer was inserted. Using the same control logic as the transition-once multiplexer (only one set of selects in this case), the buffer is allowed to transition when the input control signals are stabilized. By cutting down the intermediate transitions, the amount of power used by the random logic could also be reduced.

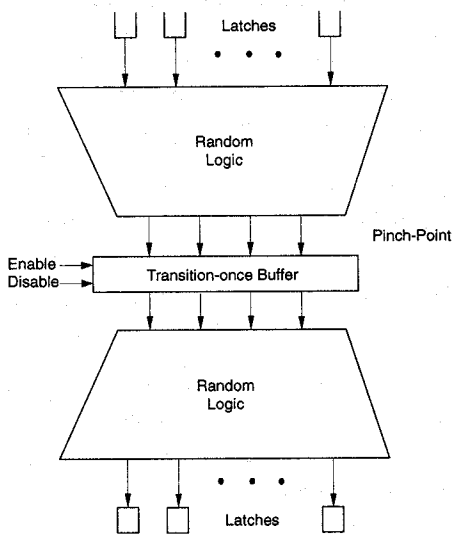

Figure 7 Transition-Once Buffer

When a pinch-point exists in the design, a transition-once buffer can isolate the downstream logic cone from the upstream cone, reducing transient node toggles.

Insertion of timed buffers is an easy way of reducing the AC toggle in random logic. The overhead is the addition of the timed selects and the buffer itself. The result is the elimination of downstream toggles due to the intermediate transitions of the previous stage logic.

TECHNOLOGY AND CORE LIBRARY

In general, macro performance was optimized to meet low power macro objectives. Quasi-differential drive latches were used to reduce the clock input loading by 80%. Low power clock splitters are used exclusively to match low power latches and enable clock gating. The ASIC library was expanded to include drive strengths with smaller device widths so power could be minimized on paths with non-critical timing.

An analog PLL was designed with low power features. Starting from an existing design that was built for higher performance chips, the maximum operating frequency was reduced from 400 MHz to 100 MHz. Clock shutdown modes specified by the chip architecture allowed the PLL to be commanded to suspend operation during chip sleep modes. Node toggling was eliminated on logic branches that are unused during a given mode of operation. DC paths were reduced or eliminated.

For the ROM, input addresses are latched only if a ROM access is requested. Next, partitioning and proper sequencing of the ROM timing events was orchestrated to minimize overlap of events that could cause DC paths or high power conditions. The word line decoder divides the 256-word lines into 32 parts of 8-word lines each. One part is active at a time. For each data output bit, only one of the 32-bit lines that could be accessed is pre charged. Finally, ROM output data is held until the next ROM read is executed, reducing downstream switching.

The multi-port SRAM used a number of techniques to minimize power consumption. Clock-gating was performed in the interface logic based on port request signals, minimizing clock tree and latch power. The array was implemented using three sequential operations to a standard six-transistor cell to reduce area. Support circuits were fully interlocked, maximizing the power/performance efficiency. To minimize array power, a NAND decode word-line system was chosen, and word-line length was kept to 64 bits, reducing bit-line pre-charge power and all associated sense amp and control signal power. Standby power was reduced, primarily by reducing sub-vt leakage of the OFF word line driver PFET devices (greater than minimum channel length was used).

Technology leverage was used to reduce voltage to core logic from 2.5 V to less than 2.0 V. This results in a power reduction of over 20%. Power shutdown circuitry is incorporated into voltage regulators. A diode is used during shutdown mode to hold core voltage to non-critical circuits.

SUMMARY AND RESULTS

We have described a methodology and design methods for creating a low-power processor design. Designing for lowest possible power consumption requires attention to the entire design process, including: technology, ASIC library selection, power reduction methodology development, architectural and gate-level logic design techniques, and physical design methodology.

A complete HDL-to-gate level methodology has been described to successfully identify, track, and implement a power reduction strategy. Emphasis was placed on power reporting and analysis, and automatic low-power synthesis capabilities.

Several innovative design techniques were described for dynamic power reduction, including register typing for control logic clock gating, a transition-once multiplexer to reduce data-path node toggles, and a pseudo-microcode approach to processor instruction decoding to minimize overall instruction stream dependent power.

Acknowledgments

The authors would like to thank the following people for their contributions to this paper: John Adams, Dave Blum, Rob Busch, Dr. Ken Goodnow, Roger Gregor, Gary Koch, Manabu Ohkubo, and Chris Ro.