POWER MANAGEMENT TECHNIQUES SURVEY
With emphasis on the M2

10/21/95
Pete Foley

1 INTRODUCTION:

1.1 Goal: Power Efficiency as a First Class Design Citizen

I have conducted a quick survey of some of the literature on low power design techniques, and a summary is presented below.

Power management is fast becoming a crucial design component of the Mpact processor family, and as such it deserves to be elevated to “first class citizen” design status irrespective of whether the target platform is the desktop or the portable.

The goal of this survey is to help foster an ethic of power efficient design. As Mark Horowitz put it "Probably the most important low-power design method is simply to make low power a key objective in the design process"[5].

1.2 Green Computing

“Green computing” design mandates require systems where power consumption can be explicitly specified, with performance the dependent variable.

A computer on the desk of every person in even modest sized corporations can consume a whopping amount of power (don’t forget the extra air conditioning load to cool the building!). The EPA estimates that computer equipment is responsible for 5-10% of all commercial electrical power consumption in the US.

1.3 M2 in Portables

One of the principal goals of the M2 is its use in portable products. The M1 was not designed with low power systems in mind, so emphasis has shifted from utilizing the M1 in low power systems to using the M2. Therefore extremely low power quiescent or “sleep” modes are a new requirement.

1.4 Plastic Packaging Key to Pricing Model

Plastic packaging is a crucial component of the Mpact family cost structure. Over 50% of the cost of a Pentium is the packaging, so that to package the M2 in a Ceramic or PowerQuad or Aluminum Nitride package moves the Mpact into an unacceptable price domain. New packaging technologies such as BGA will allow some increases in power dissipation, but current data indicates that plastic BGA will offer only modest thermal improvements over PQFP with a heat spreader.

1.5 Trends Towards Greater Power Consumption

1.5.1 Active Power

The current M2 design center incorporates a 220Mhz RAMDAC, increases the clock rate, adds a second RAMBUS channel, adds a significant number of gates for 3D graphics hardware pipeline and other functional improvements, perhaps increases the PCI bus clock rate, and increases the capacitive loading (particularly for the clock) via superpipelining. In order to maintain a worst case power dissipation in the 3W range, despite the power advantages offered by .35 micron technology, the M2 designers will need to consider power management issues at the ground floor level.

1.5.2 Quiescent Power

Portables require instant wakeup capabilities, the system cannot be restored from disk. Hence the current quiescent power dissipation incurred by the M1 with its active RAMBUS channel style of RDRAM refresh must be addressed in M2, especially if dual RAMBUS channels are used.

Very low power means of monitoring the system for wakeup events must also be implemented if the M2 is to be competitive in the battery operated market.

2 WHERE ALL THE JUICE GOES

Power consumption in CMOS systems is given by the equation below:

$$Power = \left(\sum_{i=1}^{N} p_i \cdot C_i \right) \cdot V \cdot V_{dd} \cdot F + (I_{sc} + I_{leak}) \cdot V_{dd} + P_{glitch}$$

Where V is the logic output voltage swing, V_{dd} is the supply voltage, F the clock frequency, N the number of nodes in the circuit, p_i is the probability that the ith node toggles in a clock cycle, and C_i is the load capacitance of the ith node. I_{sc} is the short (or shunt) circuit current of the complementary CMOS device as the input(s) transition through the region where both PMOS and NMOS circuits are on. I_{leak} is the leakage current, which is process dependent and is usually ignored, and P_{glitch} is the power due to hazards and their propagation through the circuit.

3 PARAMETER OPTIMIZATION

All of the parameters of the Power equation are candidates for optimization:

V & V_{dd} : These are the most productive parameters to optimize because of the quadratic dependence of power on voltage when outputs swing rail to rail. Tactics include simply lowering overall voltage or creating separate voltage islands on the chip for non speed critical logic with level shifters inserted where appropriate.

F : Frequency can be modulated on a regional basis, or on a global basis, depending on the actual compute needs of the circuit at any given time. This was the parameter generally optimized by first generation portable core chipsets that simply measured the case temperature of the Pentium and throttled the clock frequency accordingly.

C_i : Interconnect load capacitance can be lowered through better floorplanning and through technology tricks such as air bridges. Load capacitance can be lowered through better synthesis optimization. Use of hierarchical structures, such as additional cache levels can effectively lower the average load capacitance seen by particular operations.

Techniques such as charge sharing to approximate so called “adiabatic computing” techniques can also be thought of as lowering effective load capacitance.

p_i : Node toggle probability is one of the most fruitful areas for optimization. This can take the form of:

a) Gating clocks
b) Minimizing state “transition densities” in state machine control logic.
c) Isolating (guarding) inputs of logic blocks when evaluation by that logic block is not needed.

I_{sc} : Short circuit current can be minimized through process choices such as threshhold voltages, and by making careful control of logic transitions throughout the circuit. Elimination of any signals that transition too slowly is key.

I_{leak}: Is primarily the sub-threshold source-drain current due to carrier diffusion between the source and the drain when the gate-source voltage has exceeded the weak inversion point, but is still below the threshold voltage. This parameter generally becomes worse (increases) as the supply voltage is dropped.

P_{glitch} : Glitch power (caused by arrival time skew of input signals to combinational logic inputs) can account for as much as 20% of power in combinational logic blocks. Total power consumed is strongly dependent on circuit topology, and average path length of propagating hazards [27]. A static 8-bit ripple carry adder with a uniformly distributed set of random input patterns will typically consume an extra 30% power [4].

4 GENERAL REFERENCES

Some good sources include (a * indicates we have these references):

[1] *The best reference text on the subject is, *Low Power Digital CMOS Design*, by R. Brodersen, Kluwer Academic Publishers, 1995.

[2] **Proceedings of the 1995 International Symposium on Low Power Design*

[3] **Proceedings of the 1994 Symposium on Low Power Electronics*, San Diego CA, Oct. 1994

[4] *A. Chandrakasan, et. al., "Low-Power CMOS Digital Design", in *IEEE Journal of Solid-State Circuits*, April 1992

References to most of the individual papers are provided at the end of this document.

5 PAPERS

I try to steer away from papers that are clearly not applicable to our design space (I am not totally successful though). An example is a dynamic logic style such as Clocked CMOS logic or Pulsed Power Supply CMOS [17].

The papers "reviewed" below are loosely grouped by parameter being addressed.

5.1 Lowering Voltage

When voltages are reduced, noise margins can be relaxed because of the reduced switching currents and resulting power rail bouncing. However, there is a limit to the reduction in power supply voltage because of the increase in sub-threshold circuits. The minimum threshold voltage for CMOS at which the leakage current is acceptable is about .2V. Supply voltage choice would then be depend on the noise margins desired.

5.1.1 Logic Block Decomposition/Parallelization

On of the more interesting approaches is to introduce more concurrency into a circuit to speed it up and then to reduce the voltage until it realizes its original required throughput. This trades area for power, but power savings can be significant for small area increases because of the quadratic dependence of power on voltage[10].

In [26], the authors trade space (but typically not that much space) for power by decomposition and parallelization of logic blocks in order to reduce the operating frequency of the sub-blocks, thus allowing those sub-blocks to operate at a lower voltage. The authors give examples of a parallelized program counter, shift register, LFSR, etc. An example for a shift register is shown below:

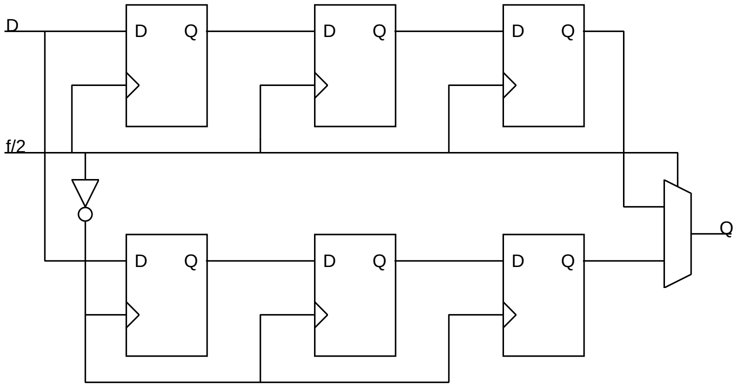

Figure 1: Parallelized Shift Register

5.1.2 Voltage Clustering

Horowitz [20] has proposed a voltage scaling technique that clusters logic in voltage domains depending on the needed speed. If enough logic can be clustered together, the overhead of the inter-domain voltage translation logic can be made acceptable.

5.2 Minimizing Node Toggle Probability

5.2.1 FSM Power Minimization:

5.2.1.1 Self Loop Detection

One technique detects self-loops in Mealy or Moore type FSM's (where the next state and the the outputs do not change on the next clock edge). In this condition the clocks to the FSM can be gated off. This is analogous to the use of sensitivity lists to trigger Verilog "always at" blocks to avoid boolean evaluation at every clock.

5.2.1.2 Clock Domain Partitioning

Another scheme proposed by Papachristou [21] et. al.partitions logic into clock domains where each domain is oparated at frequency *f/n* from one of *n* non-overlapping clocks derived from the main clock. The idea is to force logic in each clock domain to make transitions only during the clock period dedicated to that domain.

5.2.1.3 Minimizing State Transition Densities

"The state assignment problem, for low power dissipation, should be reformulated as the problem of minimizing the number of state bits switching per state transition. Any algorithm which uses average switching as its cost function can be classified as an MWHD (Minimum Weighted Hamming Distance) algorithm."[28]

Roy and Prasad in [28] presented a MWHD algorithm called SYCLOP. Their algorithm tries to minimize the logic area and what they define as the *transition density* of the internal nodes of the FSM. It is interesting that the function they attempted to minimize was complex enough that a simulated annealing algorithm was used to derive the state assignments.

5.2.2 Grey Code Addressing

"On average, using Gray code adressing reduces by 33% and 12% the bit switches in the instruction and data (cache) address buses, respectively."[24]

"When we apply Gray code addressing and cache sub-banking on a 32K-byte four-way set associative cache with 32-byte cache lines, the overall cache energy consumption is reduced 76%".[24]

5.2.3 Avoidance of Syncronously Clocked Counters

"The (Newton System) ASIC uses ripple clocked counters for all event timers." [7]

5.2.4 Misc

IBM, in the PowerPC4xx embedded controller family, used the following techniques to reduce power dissipation[25]:

1) "Operand Isolation". The isolation of sections of circuitry from "seeing" changes on their inputs unless they are expected to respond to them.
2) Extensive use of NFET only transfer gates to allow the use of single phase clocks and controls.
3) Dynamic on-chip clock activation at the register level.

5.3 Minimizing Capacitance

CMOS logic has rapidly caught up to bipolar logic in power dissipation for high frequency operation because of the charging energy required for the increasing interconnect (especially interconnect!) and gate capacitance. As John Moussouris pointed out at the Microprocessor Forum, superpipelined processors make better sense for bipolar processors from a power standpoint, while superscalar processors are more power efficient for CMOS. However, superscalar processors are significantly more complex to design that superpipelined processors.

5.4 Minimizing Frequency

Dynamically throttling the processor clock frequency is the most popular power management technique implemented in today's x86 core chipsets. It reduces power at the expense of throughput.

In portable systems, where the x86 can exceed thermal limits of the packaging in the fanless environment, the extent to which the processor is throttled down is often governed, in part, by a thermal sensor attached to the x86 case.

Clock throttling will also be an important power management tool for the M2, and is one of the rationale for decoupling the core clock from the RAMBUS channel clock. And should the M2 be used in a situation where clock throttling is critical to thermal control, then *activity monitoring* logic should be included in the M2 design which accurately portrays the activity of the processor and I/O so that power can accurately be determined without the need for a case thermal sensor.

5.5 Minimizing Glitch Power

Spurious transitions and their associated power consumption are inherent disadvantages of static logic design. Due to finite propagation delays from one logic block to the next, critical races of dynamic hazards are inherent to static logic designs and hence a node can have multiple transistion in one single clock cycle before settling to the correct logic level. This unnecessary switching activity can consume in excess of 30% of the circuit energy required to perform the computation.

5.5.1 Register Migration

One approach to minimizing glitch power is to retime the circuit through the migration of registers through the design. Ordinarily, such retiming is done to improve the speed of curcuits, but in [11], it is applied to reduce the switching frequency of circuits. For arithmetic circuits, which show high glitching, this technique reduced power by 20%.

5.5.2 Self Timed Circuits

Another technique proposed by Ko [13] to minimize these glitches is to introduce self timed circuits to prevent glitch propagation until the circuit has settled.

6 OTHER IDEAS

Below are just a few of many possibilities. All ideas are welcome!

6.1 Hardware Screen Saver

Send a pattern out of the Display FIFO without fetching out of RDRAM.

6.2 Peripheral Activity Timers

Use timers to generate interrupts that control powering up or down I/O subsystems. This is used extensively in the Opti 465MV chipset. The timer design could be of the ripple-carry clock type to further reduce power dissipation.

6.3 I/O Trapping

Power down peripheral subsystems where applicable until an I/O access occurs, at which time the system traps and powers up the appropriate subsystem(s).

6.4 RAMBUS DLL/PLL Spin-down

Provide mechanisms to spin-down the RAMBUS DLL/PLL with the appropriate OS hooks and hardware interlocks to handle the 1us spinup time. This is a must if M2 is to be competitive in the battery powered market.

6.5 RDRAM Refresh

Use the SIO lines to refresh RDRAM instead of spinning up the DLL/PLL's and initiating transactions on the bus. This is a must if M2 is to be competitive in the battery powered market.

6.6 Compressed Framebuffer

Since display refresh is such a high duty cycle activity, if refresh data could be moved across the RAMBUS channel and through the M2 into the display refresh FIFO in compressed form, and then expanded just prior to being fed to the RAMDAC LUTs, considerable power might be saved, especially if the decompression algorithm is relatively simple.

In order to eliminate decompression logic at the RAC interface when non-refresh type accesses are desired on the framebuffer data (2D/3D acceleration, for example), memory might be traded for power by implementing a *shadowing framebuffer*, where a non-compressed copy of the framebuffer is kept for all accesses other than display refresh. Use of a shadowing framebuffer would mitigate any concerns about compliance testing or of data degradation after multiple arithmetic operations. Because rapid random read/write access to the shadowing framebuffer is not important, much more aggressive compression techniques can be used, thus reducing the area penalty of the shadowing framebuffer.

Dirty framebuffer contents could be compressed and moved into the compressed (refresh) framebuffer in a lazy fashion.

6.7 Memory Organization in Multi-RDRAM Systems

In multi-RDRAM systems, organize code and data in such a way so as to allow one or more RDRAM's to be powered down while the other(s) remain active.

For example, in office productivity applications where there is a lot of think time, if the framebuffer can be limited to a subset of the RDRAMs, the remaining RDRAMs can be powered down until the next "keystroke".

6.8 Independent Clock Frequency Domains

Decouple the RDRAM from the M2, and provide enough buffering to allow the core to be operated over a wide frequency range. This would allow dynamic tuning of the core clock frequency using the on-board PLL/clock synthesizer to throughput needs.

6.9 Moving Power Out of the Chip (the power shell game)

These ideas generally decrease on-chip power at the expense of increase system power, but may be quite desirable if it means staying within a plastic package thermal budget. Examples include:

a) Moving the RAMDAC's RS-343A doubly 75-ohm terminated RGB driver off-chip.

b) Since clock power accounts for 30-50% of total power consumption in modern CMOS processors (MIPS-X is 30%, DEC Alpha is 50%), move the primary/first stage clock driver off-chip. One of the presenters from DEC at the Microprocessor Forum indicated this was going to be done for a future version of the Alpha architecture.

6.10 LoopBack

In CMOS systems, never use tri-stated buses with pullups. Whichever bus master drove the bus last must continue to drive the bus with the last value driven.

In a system where this may not be entirely practical, or a system with a single bus master such as a microprocessor bus, then after the processor does a read, it should turn around and drive the read value onto the bus (hence the term "loopback") until a new master is granted, or until the next read. This technique was used by the Hobbit microprocessor in early Newton architectures.

6.11 On-chip State Tagging

Could we tag the SRAM contents with thread ID's, so that we can minimize the contents flushed to RDRAM on context switches? Or would the extra power overhead of tag compares more than compensate?

7 SOFTWARE OPTIMIZATIONS

7.1 Event Driven OS

"The (Newton) system is entirely event driven. There is no concept of a spin loop within the OS. Whenever the processor must wait for a slow peripheral, and there is not enough time to make a context switch, or there are no other tasks waiting to execute, the OS halts the ARM processor."[7]

7.2 Small Efficient Kernel

Keeping the kernel as small and efficient as possible is an important contributor to low power consumption. Fortunately, we are way ahead of many other players in this area.

7.3 Retargeting Compiler Cost Functions

The cost function used in most compilers and code generators is the number of execution cycles, but this can be changed to energy costs[8].

7.4 Register Allocation Strategies

"Presence of on-chip caches with high hit rates and 1 cycle access time may make register allocation less important from the point of view of performance, but it is still very important from the point of view of energy, since the average power per cache access is much higher than accessing a register"[8].

7.5 Predictive Powerdown Strategies

At the OS level, using predicive techniques to estimate future user behavior and judging when to powerdown various subsystems can be benificial, I remember reading an article where a 15% power savings was claimed for a notebook computer. Because of the M2's efficient kernel and tight coupling to the peripheral hardware, powerdown strategies of very fine granularity are possible.

8 ADIABATIC COMPUTING

Computation while consuming near theoretical minimum power via charge recovery is somewhat in vogue. It does not really apply to our design space because of the performance restrictions incurred, an analysis by Horowitz [19] showed that charge recovery techniques are only advantageous at very low operating rates, and that voltage scaling will be more energy efficient for the first two orders of magnitude in performance and over three orders of magnitude in power.

However, one of the best (at least clearest) examples of the breed is presented by Hahm [12]. For charge sharing among capacitive loads that are large, adiabatic computing techniques can be benificial. But because of the power consumed in the charge sharing control logic, the load capacitance must be about 25 times the size of the gate capacitance of a minimum sized inverter before significant power savings are attained. Data buses are good targets for this technique.

9 CACHE DESIGN

In typical processors I-caches consume more power than D-caches since loads and stores typically constitute less than 25% of executed instructions. Hence a unified cache is generally a bad idea from a power standpoint[23].

For typical RISC processors, both data and instruction caches designed in static logic consume less power if they are set associative as opposed to direct mapped. For I-caches, the higher the associativity, the better. For D-caches, and associativity of 2 is best. Increasing I-cache line size consumes more energy.[24]

Cache prefretch buffers which are used to reduce branch penalties by allowing the I-pipe to be fed while the branch target is fetch are a bad from a power consumption standpoint. “A small prefetch buffer that holds only four instructions has a very detrimental effect on the power cosumption of the I-cache since the number of instructions fetched from the I-cache increases by at least 60% . . .In low power processors, instructions should be fetched on a cycle-by-cycle basis from the I-cache”[23].

Write-back caches are more energy efficient that write-through caches.

In an interesting paper by Ramesh Panwar and David Rennels of UC Los Angeles, they present an interesting scheme to reduce the cost of cache directory lookups and tag compares for I-caches (these typically consume 25% of I-cache power).[23] Since M2 is using a direct mapped cache, this scheme is not really applicable, but it is still interesting!

10 ISA MODIFICATIONS

A controversial technique is to explicitly add state to the instruction stream to control power (clock gating, powering on/off, etc). A strength of this technique is that it exposes the power management problem to the compiler. A drawback is the decrease in code density and the power burned fetching the extra state from RAM. A less aggressive technique would be to add the power management data to the I-stream at the instruction decoder.

An example of this kind of technique was mentioned in a paper on a “low power” 68040. "The processor includes LPSTOP in the intstrucion set. When the processor decodes LPSTOP, it shuts down in one cycle to zero power dissipation (excluding leakage)."[6]

11 MISC

11.1 Low Power ALU and Adders

There were a number of papers dedicated to low power ALU and Adder designs. In order to use static logic, the emphasis was to use appropriate logic families, such as Double Pass Logic (DPL)[14], and Push-Pull Cascode Logic (PPCL)[15]. Also emphasized was optimal sizing of transistors using algorithms such as those based on "A Theory of Logical Effort”, by I. Southerland and R. Sproull[16].

11.2 Watchdog Logic

To realize very low quiescent power deep sleep modes, which is required for the battery powered market, watchdog logic is often implemented to watch for external events (external interrupts, external DMA requests, PCI access, modem ring detect, etc) and for internal events (such as timer interrupts). The watchdog logic is then responsible for the wakeup and powerup sequencing of the system, which may include spinning up the on board clock synthesizer PLL and a host of other tasks. Such watchdog logic is typically operated off of a 32Khz watch crystal, and utilizes power saving techniques such as ripple carry clocked counters.

12 REFERENCES (papers)

(a * indicates we have these references)

[5] *M. Horowitz, et. al.,"Low-Power Digital Design", *Symposium on Low Power Electronics*, Oct. 1994

[6] *T. Briggs, et. al., "A 1 Watt 68040-Compatible Microprocessor", *Symposium on Low Power Electronics*, Oct. 1994

[7] *M. Culbert, "Low Power Hardware for a High Performance PDA", *Symposium on Low Power Electronics*, Oct. 1994

[8] *V. Tiwari, et. al., "Compilation techniques for Low Energy, An Overview", *Symposium on Low Power Electronics*, Oct. 1994

[9] *S. Wuytack, et. al., "Global comm. and memory optimizing transformations for low power systems". In *Interntaional Workshop on Low Power Design*, April 1994.

[10] R. Brodersen, et. al., Hyper-lp: "A system for power minimization using architectural transformations". In *Proceedings of the International Conference on Computer-Aided Design*, pages 300-303, 1992.

[11] J. Monteiro, et.al., "Retiming Sequential Circuits for Low Power", In *Proceedings of the Int'l Converence on Computer-Aided Design*. pages 398-402, Nov 1993.

[12] *M. Hahm, "Modest Power Savings for Applications Dominated by Switching of Large Capacitive Loads", in *Interntaional Workshop on Low Power Design*, April 1994.

[13] *U. Ko, et. al., "A Self-timed Method to Minimize Spurious Transitions in Low Power CMOS Circuits". In I*nterntaional Workshop on Low Power Design*, April 1994.

[14] Makoto Suzuki & others, "A 1.5ns 32b CMOS ALU in Double Pass-Transistor Logic" 1993 ISSCC.

[15] A. Guyot, et. al., "A way of building efficient Carry-Skip adders", *IEEE Transactions on Computers*, October 1987, Pgs 1144-1151.

[16] Ivan Sutherland & R. Sproull, "A theory of Logical Effort", SSA #4759, Copyright 1987, Sproull and Associates, Inc.

[17] *T. Gabara, "Pulsed Power Supply CMOS- PPS CMOS", In *Interntaional Workshop on Low Power Design*, April 1994.

[18] C. Seitz, et. al., "Hot -Clock NMOS", *Proceedings 1985 Chapel Hill Conf. on VLSI Comp. Sci. Press*, pp 1-17.

[19] *M. Horowitz, et. al., "Evaluation of Charge Recovery Circuits and Adiabatic Switching for Low Power CMOS Design", In *Interntaional Workshop on Low Power Design*, April 1994.

[20] *K. Usami and M. Horowitz, “Clustered Voltage Scaling Technique for Low-Power Design”, in *Proceedings of the 1995 International Symposium on Low Power Design*, April 1995.

[21] *P. Papachristou, et. al, “A Multiple Clock Scheme for Low Power RTL Design”, in *Proceedings of the 1995 International Symposium on Low Power Design*, April 1995.

[22] A. Chandrakasan et. al, “Optimizing Power Using Transformations,” *IEEE Trans. on CAD*, Jan. 1995.

[23] *Ramesh Panwar and David Rennels, “Reducing the frequency of tag compares for low power I-cache design”, in *Proceedings of the 1995 International Symposium on Low Power Design,* April 1995.

[24] *A. Despain, et. al., “Cache Design Trade-offs for Power and Performance Optimization: A Case Study, in *Proceedings of the 1995 International Symposium on Low Power Design,* April 1995.

[25] *Correale, “Overview of the Power Minimization Techniques Employed in the IBM PowerPC 4xx Embedded Controllers”, in *Proceedings of the 1995 International Symposium on Low Power Design,* April 1995

[26] *C. Piguet et. al., “Logic Design for Low-Voltage/Low-Power CMOS Circuits”, in *Proceedings of the 1995 International Symposium on Low Power Design,* April 1995.

[27] *M. Favalli and L. Benini, “Analysis of Glitch Power Dissipation in CMOS IC’s”, in *Proceedings of the 1995 International Symposium on Low Power Design,* April 1995.

[28] *K. Roy and S. Prasad. “Syclop:Synthesis of CMOS Logic for Low Power Application”. *Proceedings of the international Conference on Computer Design,* Oct. 1992.