Microarchitectural Simulation and Control of di/dt-induced Power Supply Voltage Variation

Ed Grochowski
Intel Labs
Intel Corporation
2200 Mission College Blvd
Santa Clara, CA 95052
Mailstop SC12-303
edward.grochowski@intel.com

Dave Ayers
Enterprise Processor Division
Intel Corporation
2200 Mission College Blvd
Santa Clara, CA 95052
Mailstop SC12-502
david.ayers@intel.com

Vivek Tiwari
Intel Architecture Group
Intel Corporation
2200 Mission College Blvd
Santa Clara, CA 95052
Mailstop SC12-603
vivek.tiwari@intel.com

Abstract

As the power consumption of modern high-performance microprocessors increases beyond 100W, power becomes an increasingly important design consideration. This paper presents a novel technique to simulate power supply voltage variation as a result of varying activity levels within the microprocessor when executing typical software. The voltage simulation capability may be added to existing microarchitecture simulators that determine the activities of each functional block on a clock-by-clock basis. We then discuss how the same technique can be implemented in logic on the microprocessor die to enable real-time computation of current consumption and power supply voltage. When used in a feedback loop, this logic makes it possible to control the microprocessor's activities to reduce demands on the power delivery system. With on-die voltage computation and di/dt control, we show that a significant reduction in power supply voltage variation may be achieved with little performance loss or average power increase.

1. Introduction

Over the past 25 years, microprocessor power consumption has grown from under one watt to over 100 watts. The dramatic increase in power is a result of transistor scaling, which has produced many more transistors on a chip running at much higher frequencies [1]. Traditionally, voltage scaling has been used to reduce power to manageable levels; however, with supply voltages approaching one volt, further large reductions in voltage are unlikely.

Today power has emerged at the forefront of challenges facing the microprocessor designer. There are two distinct sets of problems - those associated with absolute power level and those associated with changes in power level. On absolute power level, it is clear that a microprocessor consuming 100W requires a power supply, voltage regulator, and power distribution network capable of supplying 100W, as well as a thermal solution (package, heat sinks, and fans) capable of dissipating the resulting heat. Such components are costly and cannot be expected to scale to higher power levels as transistor dimensions shrink. Changes in power level are also problematic - a hypothetical 100W microprocessor running at 1.0V draws 100A. The voltage regulator and power distribution network must maintain the supply voltage to within +/-5%, meaning that no more than 100mV peak-to-peak ripple can be tolerated regardless of what the microprocessor and the software its running do. The power distribution network must have sufficient capacitance, and small enough inductance and resistance, to maintain the supply voltage to within 100mV even though the microprocessor's supply current may change dramatically within a few nanoseconds. This latter problem is referred to as the *di/dt problem* after the definition of inductance $V=L*di/dt$. V is the voltage across an inductor of value L when subject to a change in current di/dt.

2. Microarchitecture Simulators

There are many existing microarchitectural simulators: SimpleScalar [2], SMTSIM [3], etc. These simulators work by simulating the flow of instructions through the microprocessor's pipeline. The simulator contains an implementation of the microprocessor's pipeline and control logic as well as an architectural simulator needed

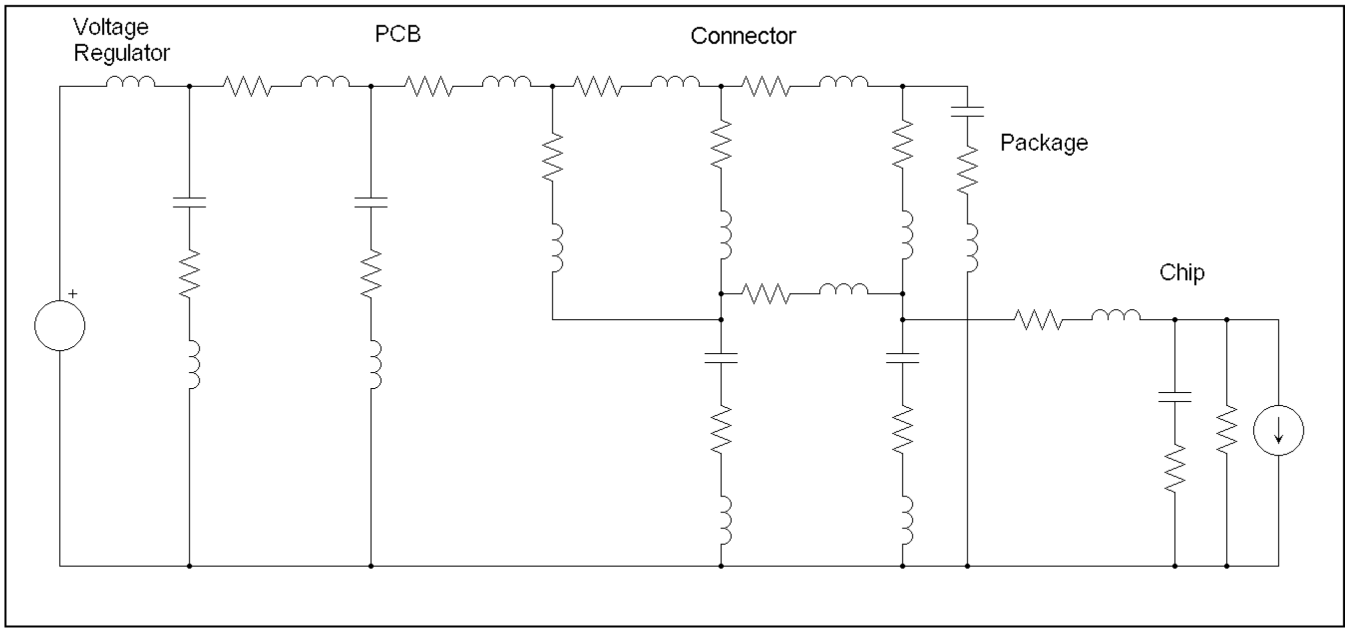

Figure 1: Electrical Model of Power Distribution Network

to execute the program. The simulator determines how many clocks are required to execute a given benchmark, and produces statistics such as pipeline stalls, cache misses, mispredicted branches, etc.

Recent work has shown how to augment a microarchitectural simulator with power computation [4]. The power simulator makes use of a feature of modern microprocessors – extensive clock gating. Each functional unit is equipped with a circuit that turns on the clock when the unit is active and shuts off the clock when the unit is inactive. This technique minimizes power consumption in inactive units but results in large variations in overall power levels that depend on the software being run. In a very power-conscious design, the clock gating may be done with extremely fine granularity - unit by unit and pipestage by pipestage - resulting in a large number of clock gating signals. For every clock the simulator knows which units and pipestages are active, and can compute the total power consumption during that clock by simply adding up the active power and idle power of blocks that are *on* and *off* respectively. Although usually thought of as a power simulator, the simulator can readily compute the total supply current and multiply current by a constant voltage to produce power. *Current* is the interesting quantity, as we'll see in the following section.

Several such simulators have been built for internal use at Intel. The active and idle currents for each block are based on low-level circuit simulations or estimates. The accuracy of these simulators has been determined to be sufficient for pre-silicon comparisons between different microarchitectural design points across a range of software applications.

3. From Current Simulator to Voltage Simulator

Once a current simulator has been constructed, the power supply voltage may be computed based on an understanding of how the power distribution network responds to new demands for current. Figure 1 shows an electrical model of the power distribution network for a high performance microprocessor. The model includes decoupling capacitors on the die, in the package, and in the voltage regulator, as well as the parasitic inductance and resistance associated with the package, socket, printed circuit board, and devices within the voltage regulator. The microprocessor is modeled as a variable current sink, and the remainder of the voltage regulator is modeled as an ideal voltage source. Component values are chosen to be representative of those in this application. A discussion of how to construct such a model is beyond the scope of this paper. Refer to [5] for a description of the relevant techniques.

The result of applying a 25A current step to the power distribution network is shown in figure 2. The supply voltage as seen by the microprocessor dips and *rings* due to the inductance and capacitance of the power distribution network. The voltage reaches a minimum value 25 clocks after the current step began (each clock is 0.3ns).

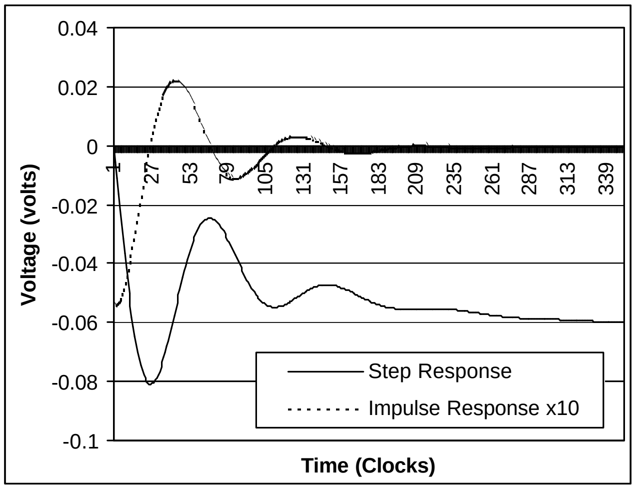

Figure 2: Step and Impulse Response of Power Distribution Network

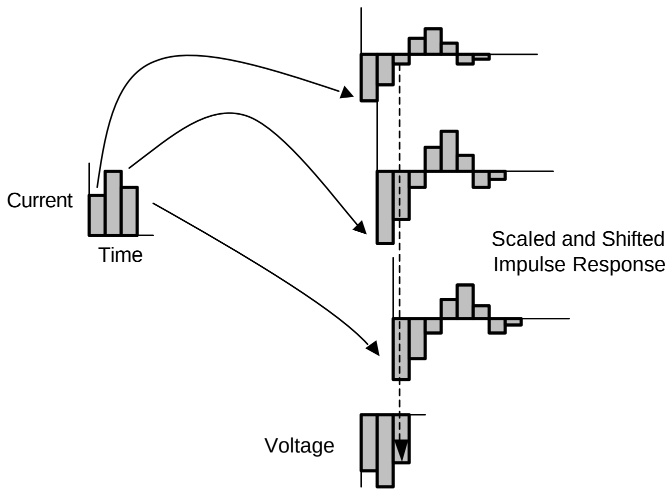

Figure 3: Voltage Computation using Convolution

The step response was determined by running a Spice-like circuit simulation on the network in figure 1. Also shown is the impulse response, which is the response of the power distribution network to a single 25A pulse of 0.3ns duration. The impulse response was computed from the step response by taking the difference between adjacent samples. For simplicity, we've considered only the high-frequency (50-100 MHz) components caused by the circuit board, connector, package, and chip. We've neglected a low-frequency (~1 MHz) resonance caused by the voltage regulator's LC filter. We'll discuss the implications of including this in a later section.

Now that we've determined the response of the power distribution network to a fixed current step, how can we determine its response to an arbitrary current waveform such as the processor's variable activity over time? At first glance, this might appear to require solving systems of differential equations. However, a much simpler method is possible based on the observation that the power distribution network is, to rough approximation, a linear network. Linear systems satisfy two properties:

1. Scaling the input by a certain amount causes the output to scale in proportion. Mathematically, f(c*x)=c*f(x) where *c* is a constant. Thus, if we were to double the amplitude of the input current step in figure 2, we would expect the output voltage drop to double.
2. The result of applying the linear function to the sum of two inputs is the same as if the function were applied individually to each input and the results summed. Mathematically, f(x+y)=f(x)+f(y). Thus, we can compute the response to a sum of two input waveforms by applying the linear function to each input individually and summing the results. This property is known as *superposition*.

The two properties of linear systems make possible simulation of power supply voltage for arbitrary currents using simple mathematics. This is done by decomposing the current waveform into a series of pulses (one pulse per clock), scaling and shifting the impulse response by the height and time offset of each current pulse, and adding together the results. The algorithm is illustrated in figure 3. Since the current simulator computes average current per clock, it is natural to use the clock as the unit of time for a pulse in the voltage simulator. Readers familiar with digital signal processing will recognize that our supply voltage simulator is computing a convolution of the processor's simulated power supply current and the impulse response of the power distribution network [6].

The results of the combined current and voltage simulator are shown in figures 4 and 5. This is a 2000 clock excerpt of a simulation of a future Itanium™ processor running the Apache web server and gzip file compression program. As one would expect, the current graph reflects phases of program execution, each phase having its own unique IPC and current levels. The voltage graph roughly follows the current graph, with phases of high IPC and high current having low supply voltage. Over the entire 20 million clock simulation the processor's supply current varies from 52A to 73A. The peak-to-peak voltage variation is 40mV with a 1.2V supply. Note that the voltage variation is well within the allowable tolerance since the Apache/gzip workload is not a worst-case di/dt pattern.

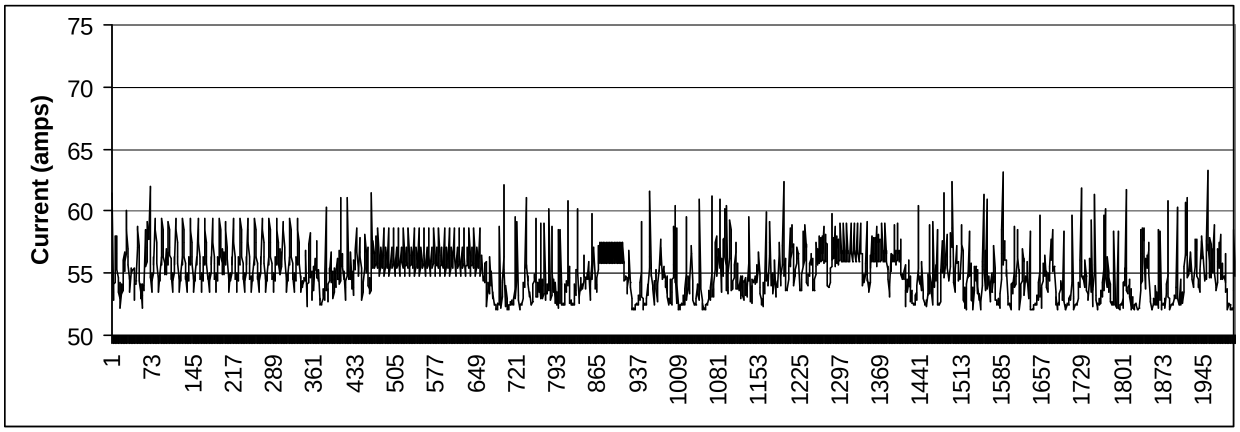

Figure 4: Microprocessor Supply Current versus Time

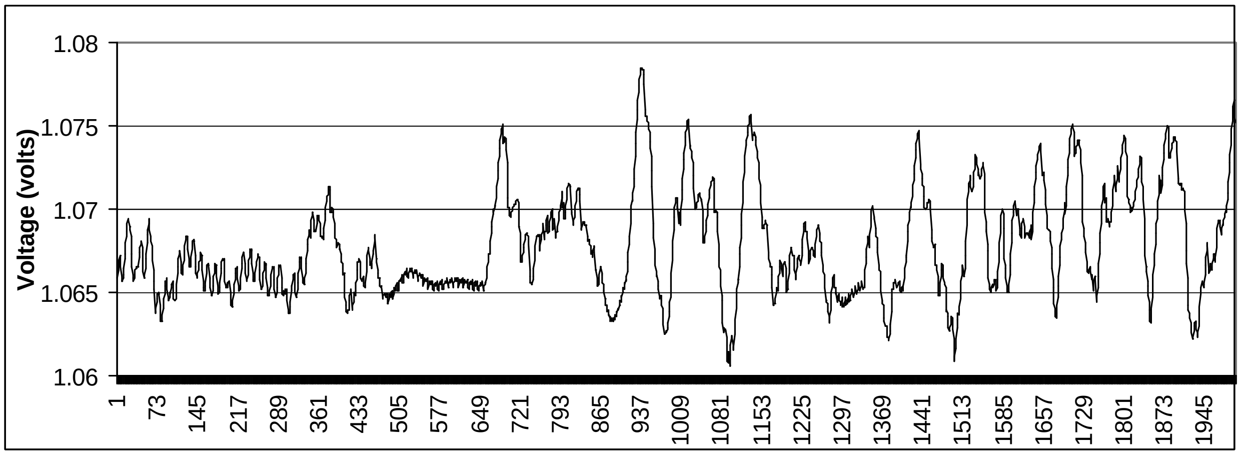

Figure 5: Microprocessor Supply Voltage versus Time

4. From Simulation to Control

So far, we've focused on simulating the microprocessor's power supply current and voltage as a function of the microarchitecture and software being run. Such simulations are invaluable to the microprocessor designer to perform design-time optimizations [7]. We now examine the possibility of controlling the microprocessor's activities in real-time to limit both absolute power and variations in supply voltage.

To understand why control of power related parameters is beneficial, consider how microprocessor power delivery systems are designed today. A microprocessor's power delivery system must be designed for the *worst-case* software that can ever be run. This is usually a program with extremely high IPC (for maximum power consumption), or a program that rapidly alternates between extremely high IPC and extremely low IPC (for maximum di/dt). We call such programs *power viruses* because they stress the power delivery system much more than normal application software. Consider now the possibility of adding on-die power computation and regulation hardware. This hardware can detect when abnormally high power demands are made by software and slow down execution of that software to reduce power, or artificially inject activity to raise power in order to keep supply voltage within preset limits. With such controls, the microprocessor and its power delivery system need only be designed to meet the needs of *typical* software, with the control hardware ensuring that the typical values are never exceeded. The cost savings in designing for typical rather than worst-case software behavior can be substantial. The Intel® Pentium® 4 processor implements a thermal monitor to limit die temperature so that the processor and system thermal solutions may be designed according to the power envelopes of real applications rather than worst-case power viruses [8].

We began our research by examining shift-register approaches to di/dt control. Shift registers have been

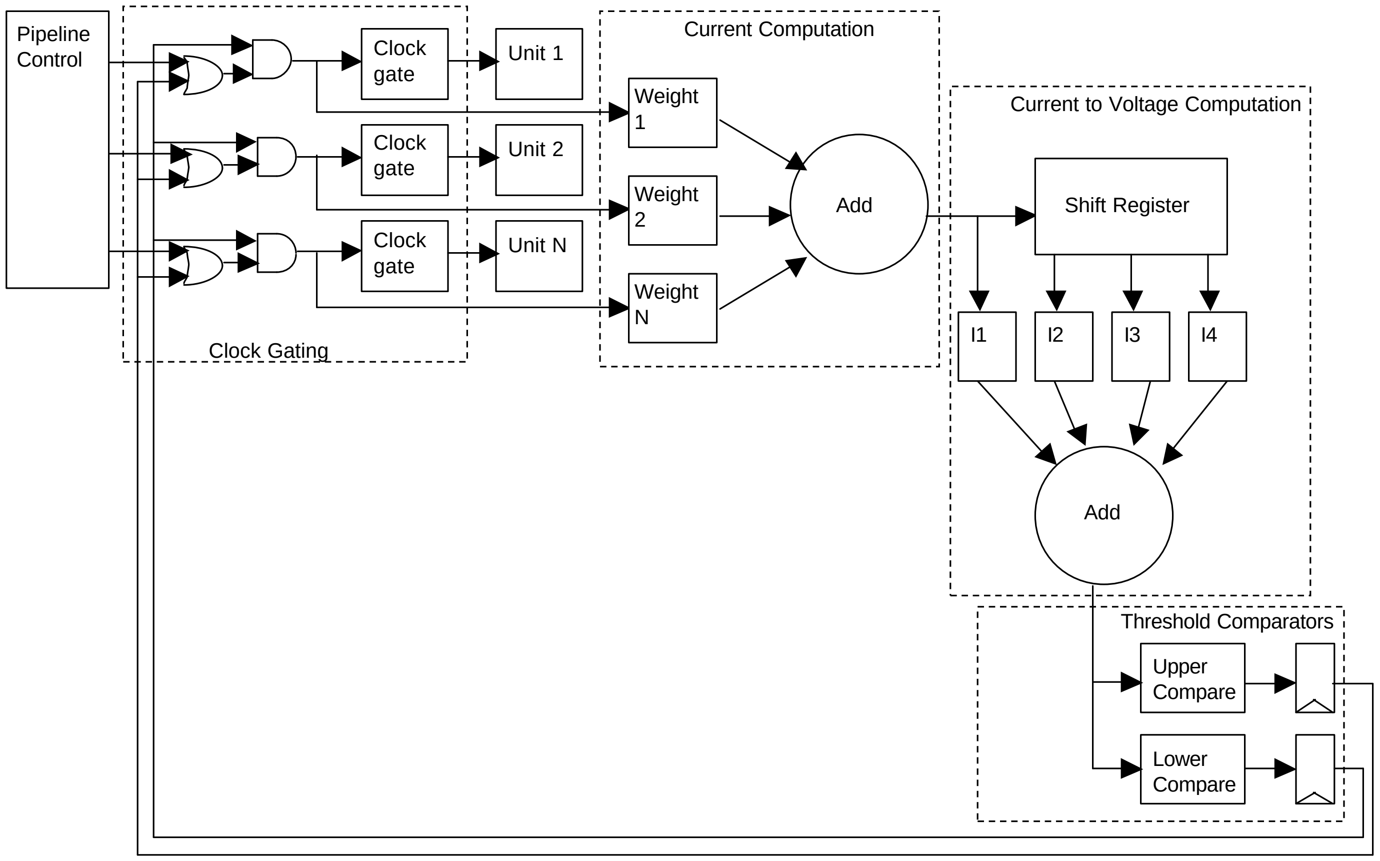

Figure 6: di/dt Controller Block Diagram

proposed as a means to reduce di/dt induced power supply voltage swings [9]. The idea is to spread out the clock gating transitions of functional units over multiple clock cycles by dividing the unit into a number of blocks and turning each block *on* and *off* at different times. This scheme is successful in spreading out large current changes over several clock cycles; however, it cannot effectively address changes in current over a time period longer than the hardware pipeline. With ever increasing clock frequencies, a different approach was needed.

We initially considered using a shift register to maintain a history of the processor's activity over time, the length being set according to the duration of the response of the power distribution network (approximately 25 clock cycles in figure 2). When a large transition is detected, such as going from a period of low activity to a period of high activity, logic connected to the shift register determines that the processor has exceeded its budget for current change, and limits the processor's future activities. This technique evolved into the shift register containing small integers (rather than single bits), the values being proportional to the supply current in that clock. The transition detection logic evolved into a weighted sum computation with both positive and negative weights. Several methods for determining the weights were experimentally tried and the best results were obtained by using weights that were proportional to the first derivative of the step response of the power distribution network. The final step was achieved when it was noted that this method was the same as setting the weights according to the impulse response of the power distribution network so that the algorithm used by the simulator to compute power supply voltage and the algorithm used by the shift register/weighted sum logic to minimize voltage variation were identical! Thus was born a systematic method to design logic that maintains the processor's supply voltage to within preset limits by computing what the supply voltage should be and controlling the microprocessor's activities to maintain those limits.

A block diagram of the complete di/dt controller is shown in figure 6. It consists of the current computation hardware that monitors the clock gating signals of each block and adds up the active current and idle current to compute the total chip current. The result is fed to a shift register that computes a weighted sum of current over time, the weights being proportional to the impulse response of the power delivery network. This is the convolution engine. The result is then fed to two digital comparators. When a lower voltage threshold is crossed, current is reduced either by shutting off the clock directly or by shutting off instruction fetch or issue causing the clock-gating circuits to shut off the

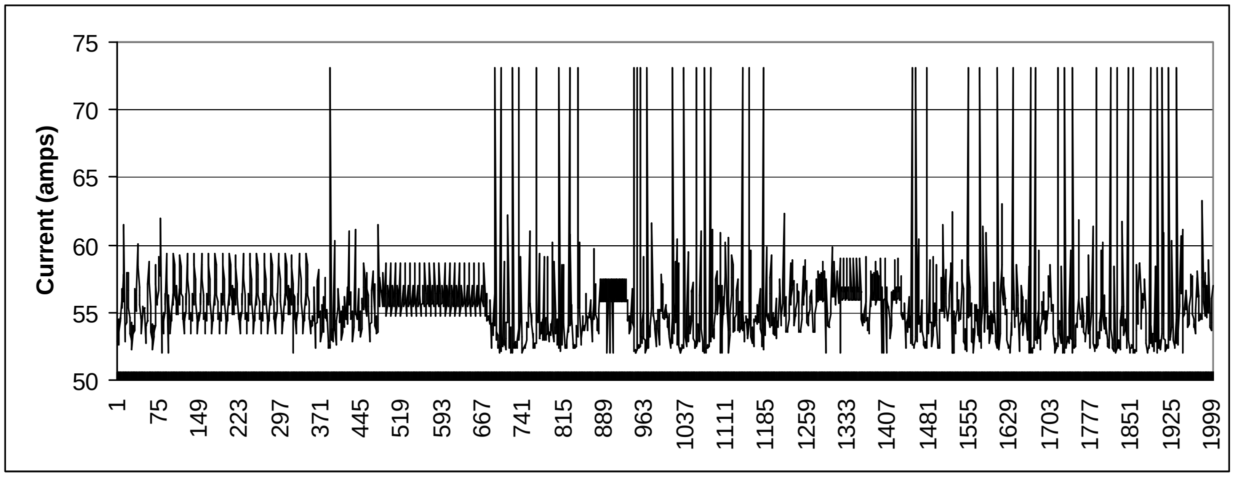

Figure 7: Current versus Time with di/dt Controller

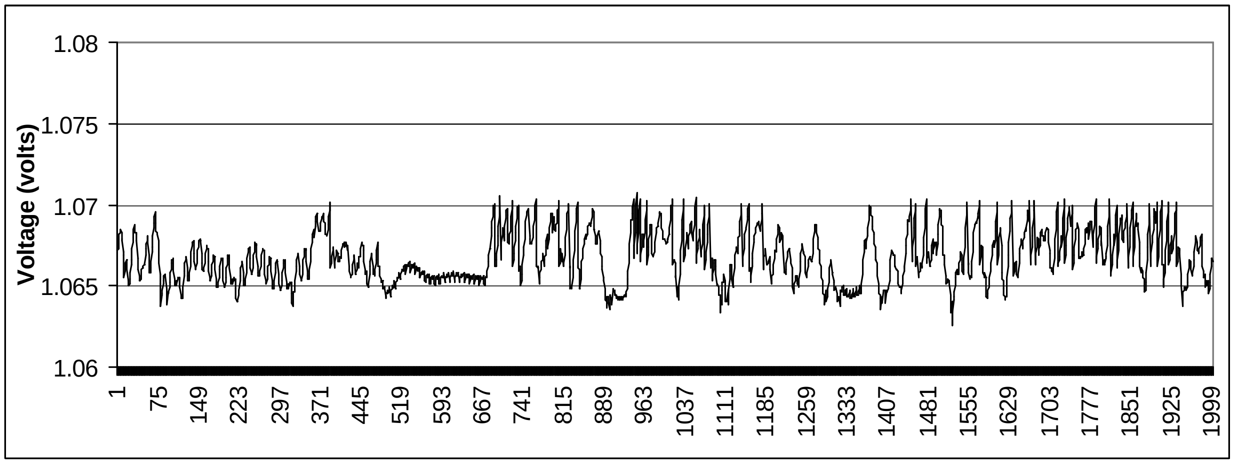

Figure 8: Supply Voltage versus Time with di/dt Controller

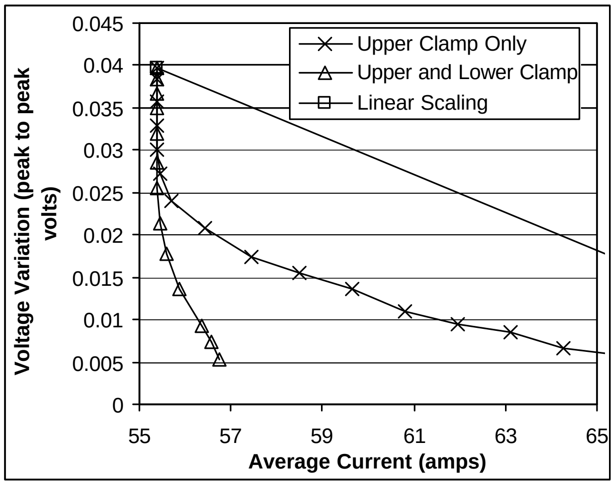

Figure 9: Supply Voltage Variation versus Current

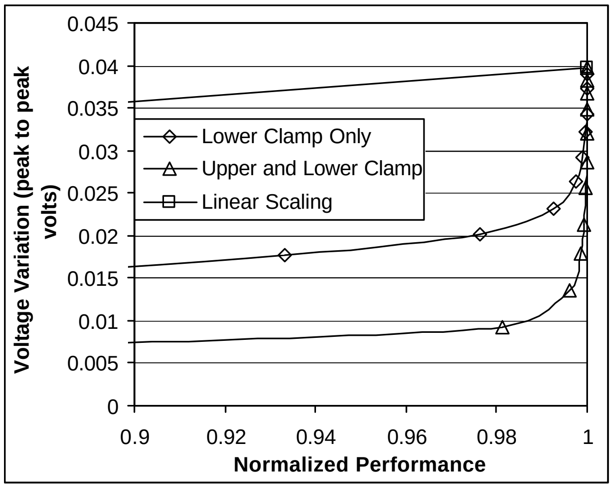

Figure 10: Supply Voltage Variation versus Performance

downstream clocks. When an upper voltage threshold is exceeded, current is increased by forcing on functional units that would otherwise be clock gated off (of course, one needs to ensure that essential state information doesn't get clobbered – the details of how to do this are beyond the scope of the paper). In between the upper and lower thresholds, the processor is allowed to run normally. The voltage computation, threshold comparison, and clock manipulation are performed every clock. As shown in figure 6, the di/dt controller is a feedback loop since the action of forcing on or off the functional units affects the computed supply current and voltage, which in turn affects the controller's subsequent actions.

The result of using a di/dt controller on power supply current and voltage is shown in figures 7 and 8. In these figures the di/dt controller's upper and lower thresholds have been set close together to illustrate their operation. Compared to figures 4 and 5, one can see that the di/dt controller has introduced sharp one clock spikes to both the maximum and minimum current levels. The spikes are a result of the di/dt controller forcing the clock on or off in response to the computed voltage falling outside predetermined limits. Inspite of the spikes, the peak-to-peak variation in supply voltage is less than the uncontrolled case because the di/dt controller has regulated the current in such a way as to avoid exciting RLC resonances in the power distribution network. Due to the contributions of DC resistance to the impulse response, the average power consumed by the microprocessor is also held to within fixed upper and lower bounds.

5. Voltage, Current, and Performance Tradeoffs

The effects of di/dt controls as described in the previous section are slower program execution and/or increased average power. This section examines the tradeoff between the amount of power supply voltage variation and slower program execution/increased average power.

Figures 9 and 10 show the current and performance effects of using a di/dt controller to reduce power supply voltage variation. Because it is possible to independently set the upper and lower thresholds, the effects of each are shown separately, and then the combination is shown. *Upper clamp only* means that only action taken by the di/dt controller is to force on functional units that would otherwise be idle when the computed supply voltage is too high. This results in a net current increase but no loss in performance. *Lower clamp only* means that the only action taken by the di/dt controller is to shut off the clock to units that would otherwise be active when supply voltage is too low, resulting in a net performance loss and also a reduction in current. *Upper and lower clamp* combines two control mechanisms. In these figures, the input is a 20M clock trace from the Apache web server and gzip file compression program. The simulator is an Intel internal tool that simulates a future Itanium™ microprocessor. The results of using the di/dt controller are compared against simple linear scaling of voltage variation against current and performance. Linear scaling assumes that supply voltage variation can be linearly reduced to zero either by reducing the delta between minimum and maximum current by artificially raising the minimum towards the maximum, or by reducing the processor's performance towards zero. The goal of the di/dt controller is to perform much better than linear scaling.

From figures 9 and 10 one can see that di/dt induced power supply voltage variation can easily be cut in half with less than 2% performance loss and less than 2% current increase on the Apache/gzip workload (neglecting the current consumed by the di/dt controller itself). The performance loss and current increase are proportionally much smaller than the reduction in supply voltage variation because the supply voltage follows a statistical distribution, and the di/dt controller can cut off the peaks without affecting the majority of clocks in between. The sharp knee in the curves is very desirable - it indicates that the di/dt controller can do its job without noticeably affecting program execution time or average current.

6. Implementation

The previous section assumed that the di/dt controller implements the current computation, a convolution with the 300 clock impulse response, and threshold comparisons, with a total latency of one clock. This is a very accurate but enormous amount of computation for one cycle latency. This section examines how relaxing the amount of computation and increasing latency affects the effectiveness of the di/dt controller.

Figures 11 and 12 show the performance and current effects of reducing the number of elements in the impulse response by truncating all elements after a given element. Satisfactory operation may be achieved using as few as the first 25 elements of the impulse response. In the next section, we'll discuss an alternative method to further reduce the amount of computation.

Latency presents a more difficult problem. The main sources of latency are the current computation logic and the convolution engine. The current computation logic adds up the current consumed by all blocks in the microprocessor and produces a result in the same clock that the microprocessor is drawing that amount of current. This might seem to be an impossible task, but

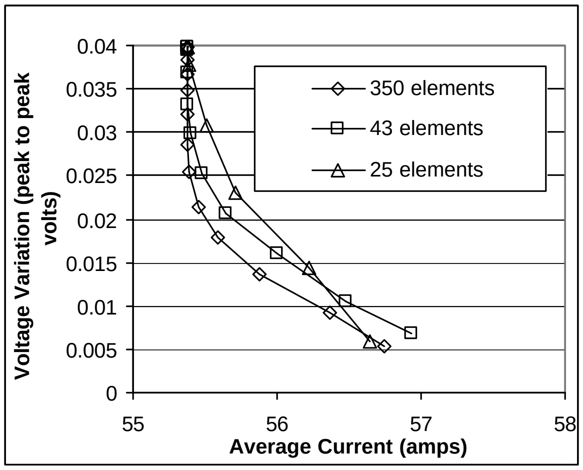

Figure 11: Supply Voltage Variation versus Current with Truncation

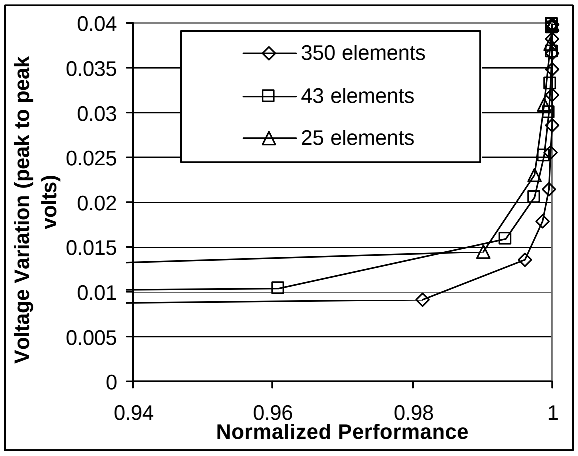

Figure 12: Supply Voltage Variation versus Performance with Truncation

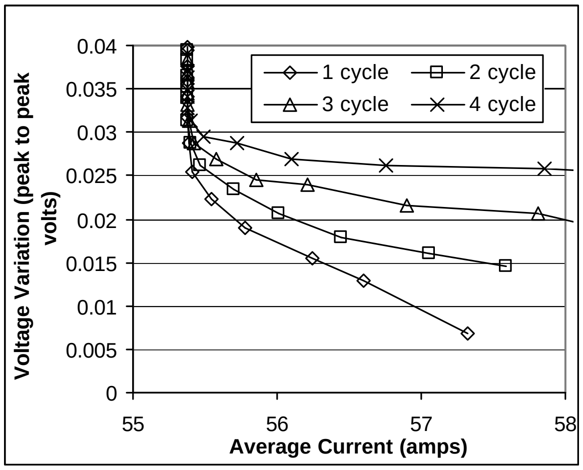

Figure 13: Supply Voltage Variation versus Current with Latency

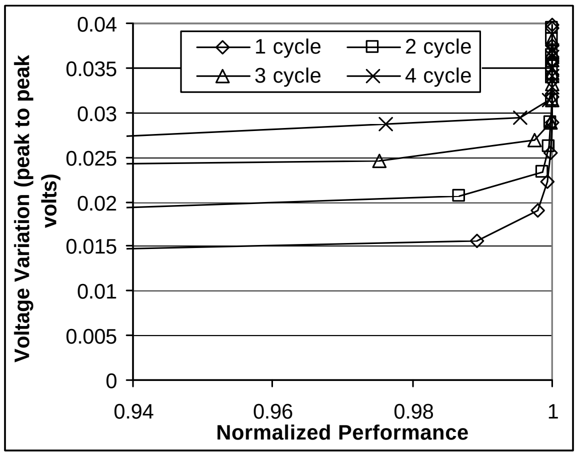

Figure 14: Supply Voltage Variation versus Performance with Latency

the current computation logic can take advantage of the microprocessor's pipeline. The supply current of blocks later in the pipeline can be precomputed at an earlier stage in the pipeline. If necessary, two computations can be performed at an early stage with a late-select choosing the correct sum according to whether the block was active or inactive. These techniques can reduce amount of logic that must operate with short latency (as viewed by the di/dt controller's feedback loop).

Latency in the convolution engine can also be addressed through pipelining. Since the convolution engine maintains a history of supply current over time, it is naturally pipelined. Only the first element of the impulse response need be computed with one cycle latency. The second element can take two clocks; the third can take three; and so on.

Figure 13 and 14 show the effects of adding latency to the di/dt controller's feedback loop. Latency is simulated by zeroing out the initial elements of the impulse response. Because the di/dt controller regulates something that is fairly fast (the "ring" in the step response), its effectiveness decreases rapidly with increasing latency, becoming ineffective with latencies greater than about 3 clocks. While the di/dt controller shares many characteristics with hardwired digital signal processors, one important difference is that achieving short latency is much more important to the di/dt controller.

Because of the need for short latency, it is expected that a realistic implementation of the di/dt controller will not likely rely on one centralized controller, but rather multiple distributed di/dt controllers, each regulating a portion of the microprocessor die. Each di/dt controller can be built alongside a microprocessor pipeline to regulate activities within its corresponding pipeline. This approach relies on loose coupling between the pipelines (through queues, for example) so that if one pipeline is momentarily stopped, incoming bits from neighboring pipelines don't fall on the floor. With a distributed implementation, we believe that a di/dt controller can be built that obtains most of the performance of the centralized controller. Clearly the performance of the distributed di/dt controller cannot be as good because each portion of the controller must operate within a fixed budget that assumes all other portions are experiencing worst-case conditions.

The di/dt controller is subject to other implementation considerations as follows:

1. The weights need to be programmable to allow for changes in operating frequency.
2. The general implementation in figure 6 is overkill – since the end result is a comparison against two thresholds, a practical implementation of the di/dt controller will use small integers with specialized logic rather than multiplier-accumulators.
3. The microarchitectural di/dt controller relies on the processor clocking much faster than the period of the "ring" in the power distribution network - it cannot control di/dt events shorter in duration than a few clocks. Adequate on-die decoupling capacitance and the pipelining technique described in [9] are required to control supply voltage swings shorter in duration than a few clock cycles.

7. Future Work

While our initial work on microarchitectural di/dt controls shows promising results, several questions remain. This section discusses possible areas for future work.

Perhaps the biggest question is “what accuracy can be achieved by the on-die supply current computation?” The current computation needs to be reasonably accurate for the di/dt controller to do its job, and furthermore it must achieve good accuracy within a limited hardware budget. If that is not possible with conventional circuit designs, we must ask the converse question: “how can a microprocessor be designed so that on-die supply current computation logic can achieve good accuracy?” Data-dependent current variations will likely set a lower bound on accuracy. The microprocessor designer may need to take steps to minimize data-dependent current variations through the use of dual-rail and differential circuit topologies, for example. The microprocessor designer may also need to formulate an error budget so that large current consuming blocks are accurately accounted for while small current consuming blocks are not considered in the current computation, but rather treated as part of the error budget.

Another question concerns the convolution engine at the heart of the di/dt controller. The convolution engine computes supply voltage from current. It is an example of a finite impulse response (FIR) filter. FIR filters are straightforward to design but not well suited to a lengthy impulse response. A recursive filter, or infinite impulse response (IIR) filter, offers the ability to handle a long impulse response with less hardware and shorter latency. A IIR filter would enable the di/dt controller to compensate for low-frequency (~1 MHz) LC ringing in the voltage regulator in addition to the high-frequency (50-100 MHz) ringing associated with circuit board, package, and high-speed decoupling capacitors. We believe the design of the filter is a good topic for future research and one that can offer significant implementation benefits.

Another area for future work is in determining the stability of the di/dt controller’s feedback loop. Feedback loops are normally never used around networks with a complex phase response due to the impossibility of making the loop stable. We’ve worked around this problem by creating a non-linear system with two thresholds – an upper and a lower threshold – that deprive the feedback loop of the gain necessary for oscillation when the voltage is in between the two thresholds. Future work should examine the relationship between the thresholds and the response of the power distribution network on the feedback loop’s stability.

A final question is to examine the tradeoff between analog and digital techniques. One may ask why build logic that computes power supply voltage when this physical quantity can be measured using analog circuits? There are advantages to each approach: the digital method can be built using the same logic gates as the rest of the microprocessor and operates completely deterministically (when fed the same input, two microprocessors will compute exactly the same result). The analog method is potentially more accurate and requires less hardware.

8. Conclusion

In this paper, we've presented a novel algorithm for simulating power supply voltage as a function of a microprocessor's activity. We've shown how the same algorithm can be implemented on the microprocessor die to reduce di/dt-induced power supply voltage variation with minimal effects on program execution speed or average power. We believe that in the future the use of such techniques will become widespread. Microprocessors containing hundreds of millions of transistors can be expected to devote a small percentage of those transistors to power computation and control.

9. Acknowledgements

The authors would like to acknowledge the contributions of David Sager and Ian Young to the development of microarchitectural di/dt controls, to Jeff Chamberlain and others for the development of the Dante simulator, and also to John Shen, George Cai, and Steve Gunther for their valuable feedback.

10. References

[1] S. Borkar, "Design challenges of technology scaling", IEEE Micro, Volume 19 Issue 4, July-August 1999, Pages 23-29.

[2] D. Burger and T.M. Austin, “The SimpleScalar toolset, version 2.0,” Computer Architecture News, Vol. 25, No. 3, Jun. 1997, Pages 13-25.

[3] D.M. Tullsen, “Simulation and modeling of a simultaneous multithreading processor”, 22nd Annual Computer Measurement Group Conference, December 1996.

[4] D. Brooks, V. Tiwari, M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations”, Proceedings of the 27th International Symposium on Computer Architecture, 2000. Pages 83-94.

[5] D.J. Herrell, B. Beker, "Modeling of power distribution systems for high-performance microprocessors", IEEE Transactions on Advanced Packaging, Volume 22, Issue 3, August 1999, Pages 240-248.

[6] S.W. Smith, “The Scientist and Engineer's Guide to Digital Signal Processing”, California Technical Publishing, 1997. Pages 107-122, http://www.dspguide.com

[7] D.M. Brooks, P. Bose, S.E. Schuster, H. Jacobson, P..N. Kudva, A. Buyuktosunoglu, J. Wellman, V. Zyuban, M. Gupta, P.W. Cook, "Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors", IEEE Micro, Volume 20, Issue 6, Nov-Dec 2000, Pages 26-44.

[8] Intel® Pentium® 4 Processor in the 423-pin Package at 1.30 GHz, 1.40 GHz, 1.50 GHz, 1.60 GHz, 1.70 GHz and 1.80 GHz Datasheet, 2001. Pages 78-79.

[9] M.D. Pant, P. Pant, D.S. Wills, V. Tiwari, "Inductive noise reduction at the architectural level”, Thirteenth International Conference on VLSI Design, 2000. Pages 162-167.