

Interconnect and Noise Immunity Design for the Pentium[®] 4 Processor

Rajesh Kumar, Desktop Platforms Group-Circuit Technology, Intel Corp.

Index words interconnect, coupling, noise, inductance, domino, cell library

ABSTRACT

For high-performance chip design in deep submicron technology, interconnect delay and circuit noise immunity have become design metrics of comparable importance to speed, area, and power. Interconnect coupling has increased dramatically due to higher metal aspect ratios with process shrinks. Reduction of transistor lengths and thresholds has led to a drastic increase in subthreshold leakage. The Pentium[®] 4 processor is Intel's fastest processor so far. It contains aggressive domino pipelines, pulsed circuits, and novel circuit families that attain very high speed at the cost of reduced-noise margins. Controlling interconnect RC delay is of paramount importance at such high frequencies. At the same time, the need for a high-volume ramp in the desktop segment necessitates high-density wiring constraints that prevent us from spacing or shielding all critical wires to manage coupling noise. All of these made the task of interconnect and noise design and verification quite challenging.

This paper describes the key innovations and learning in methodology and CAD tools. We first describe our approach to the interconnect high-frequency design problem and our silicon results. We then describe a new proprietary noise simulator (NoisePad) and our noise robust cell library, both of which enabled detailed noise design and analysis for the first time in industry and were critical to our success. Finally, inductance is a major design problem at these high speeds. Our use of a distributed power grid to manage this problem is described.

INTERCONNECT DELAY AND CROSSCAPACITANCE SCALING

With traditional process scaling, interconnect delays have not kept pace with the speedup obtained in transistors. The problem has become significant enough to require entire architectural pipe stages in the Pentium 4 processor

for interconnect communication. At the circuit level, widespread use of repeaters has become necessary. To avoid degrading interconnect resistance, the vertical dimension of metals has scaled very weakly compared to the horizontal dimension, leading to extremely high height/width aspect ratios (2-2.2). See Figure 1.

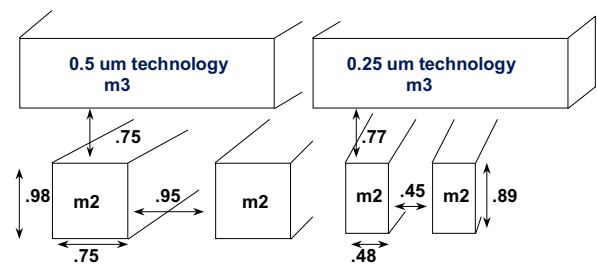


Figure 1: Wire aspect ratio scaling with technology

Nowadays, most of the wire capacitance is to parallel neighboring wires in the same layer (Figure 2), which can get routed together for long distances. This can either lead to a large increase in delay, coupling noise, or min delay problems, depending on the switching direction of neighboring wires. As can be seen from Figure 2, avoiding these delay and noise problems would involve drastically increased wire spacing or extensive shielding. Further, studies on both the Pentium[®] III and Pentium 4 processor floor plans have clearly shown that we tend to be interconnect limited for die area, which increases the penalty for spacing and shielding.

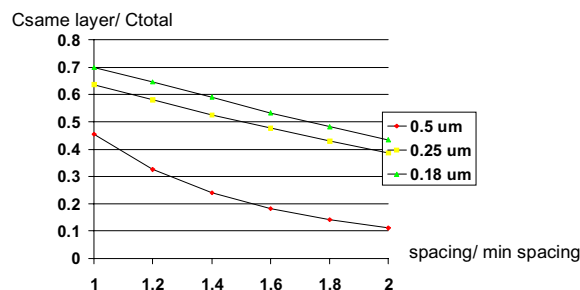


Figure 2: Coupling capacitance scaling with technology

Thus, there is a fundamental design tradeoff between a simple, robust, wiring solution employing extensive spacing and shielding vs. an aggressive solution employing short wiring with only judicious shielding leading to high density. The latter requires sophisticated CAD tools, has more risks, but ultimately is much more optimal for a high-volume product. It was therefore the choice for the Intel® Pentium 4 processor.

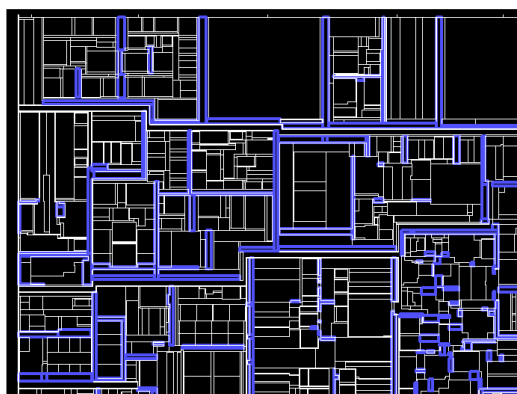


Figure 3: Dedicated repeater banks in the Pentium® 4 processor effectively form a virtual repeater grid

WIRE AND REPEATER DESIGN METHODOLOGY FOR THE PENTIUM® 4 PROCESSOR

Delay, noise, slope limits, and gate oxide wearout were all considered when drafting the guidelines for the wire and repeater methodology. Notable features were an increased emphasis on noise robustness and “pushed process” considerations for delay (repeater distance guidelines were made shorter than optimal for delay with the existing process, in anticipation of end-of-life process trending when transistors speed up a lot compared to wires). Repeater sizing, rather than best delay optimization for non-coupled wires, was picked to be optimal for noise rejection, for equal rise and fall delays, and for better delay in the presence of coupling.

Stringent limitations were put on maximum sizing of repeaters, especially in buses, to reduce power supply collapse caused by a simultaneously switching bank of repeaters. The methodology and tools allowed us to use both inverting and non-inverting repeaters. Simple length-based design rules were provided for repeaters, and further optimization was possible through internally developed proprietary tools: NoisePad, ROSES, and Visualizer (net routing and timing) analysis.

The extensive use of dedicated repeater blocks is evident in the Pentium 4 processor floorplan (with repeater blocks highlighted) shown in Figure 3. Further, the net length comparison in Figure 4 shows that although the Pentium 4 processor is a much larger chip, there are very few long nets in it compared to previous-generation chips such as the Pentium III processor. This is even more notable given that the Pentium 4 processor has more than twice as many full-chip nets as the Pentium III processor and has architecturally bigger blocks. If we compare the M5 wire segments of the Pentium III, and Pentium 4 processors, we note that 90% of the M5 wire segments of the Pentium 4 processor are shorter than 2000 microns while the same percentage of Pentium III processor wires are 3500 microns long. These short wires are a key to enabling high-frequency operation.

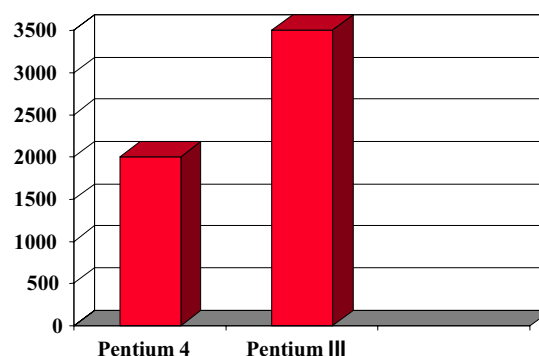


Figure 4: M5 length comparison of global wires for different processors using the same 0.18 um technology

Crosscapacitance and Density Comparative Results of the Pentium 4 Processor Interconnect

The Pentium 4 processor designers’ wiring philosophy was to allow short, tight wires. High crosscapacitance was tolerated as the price that had to be paid for dense wiring. Tolerating high crosscapacitance is necessary especially in congested areas of the chip to avoid die growth.

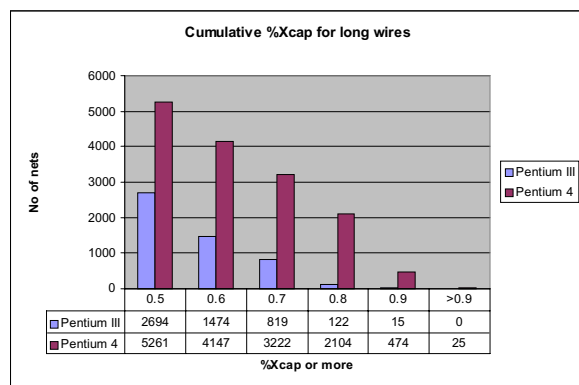


Figure 5: Coupling comparison of Pentium® 4 processor/Pentium® III processor wires

Figure 5 clearly shows that the Pentium 4 processor has significantly more wires with high crosscapacitance than does the Pentium III processor. This aggressive wiring makes additional accuracy in noise CAD tools (discussed later) even more important.

NOISE SOURCES AND TECHNOLOGY TRENDS

There exists a fundamental duality between circuit speed and noise robustness in that we can always make circuits faster by tolerating smaller noise margins. Before looking at this issue specifically from the perspective of the Pentium 4 processor, let us look at noise sources and their scaling.

Interconnect Crosscapacitance noise refers to charge injected in quiet wires by neighboring switching wires through the capacitance between them (crosscapacitance). This is perceived to be the most significant source of noise in current processes (see Figure 6). It is intimately tied to interconnect design for delay and was discussed in the previous section. Device scaling is making the problem worse due to near-end vs. far-end noise effects on resistive metal lines.

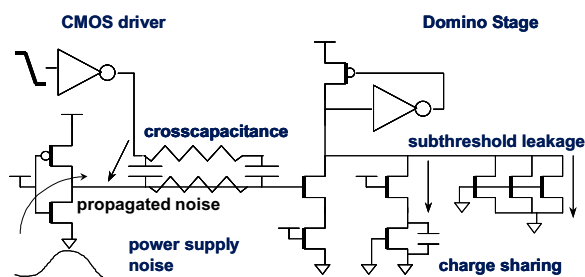


Figure 6: Various noise sources for digital circuits

Charge Sharing Noise is caused by charge redistribution between a weakly held evaluation node and intermediate

nodes in a logic stack. This primarily impacts domino nodes, weakly driven pass gate latches, and dynamic latches. The primary technology variable here is the ratio of junction capacitance to gate and interconnect capacitance. For most circuits, this noise is not getting significantly worse with new technology generations.

Charge Leakage Noise in our current processes is mainly composed of subthreshold conduction in nominally off transistors. This current can either charge/discharge a dynamic node or cause the stable state of a weakly held node to be significantly different from rails. This is mainly a concern for wide domino NOR, PLA, and memory arrays. This current increases exponentially with decreasing thresholds and is becoming very significant from 0.18 μ m onwards.

Power Supply Noise is the difference between the local voltage references of the driver and receiver, which can appear as a spurious signal to the receiver and cause circuit failure. It has both low-frequency and high-frequency components. The low-frequency component (IR drop) is managed well by flip-chip C4 packaging, which provides a very low resistance current path. For high-speed transients, the large inductance of the package return causes significant return current to flow through the on-die power grid. For simultaneous switching of wide busses, the impedances in the signal and current return path can be of comparable magnitude leading to large power supply bounce. Power supply noise is a dominant factor in the design of wide domino circuits and in circuits using contention where the AC logic level is shifted with respect to power supply rails.

Mutual Inductance noise occurs when signal switching causes transient current to flow through the loop formed by the signal wire and current return path, thus creating a changing magnetic field (see Figure 7). This induces a voltage on a quiet line, which is in or near this loop. For several signals in a bus switching simultaneously, these noise sources can be cumulative. Unlike crosscapacitance, which is a short-range phenomenon, mutual inductance can be a long-range phenomenon and hence is worse in the presence of wide busses. Faster switching speeds and wider, synchronous bus structures are making this noise very significant in current technologies.

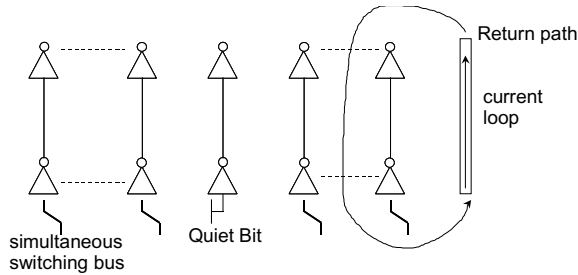


Figure 7: Mutual inductance noise from simultaneous switching on a wide bus

Inductive noise can combine with capacitive noise to cause even worse noise than shown in Figure 7. Because the analysis of inductive effects is highly dependent on layout and is quite complex, the approach is usually to design the problem out through rules rather than analyze arbitrary configurations.

NOISE CHALLENGES ON THE PENTIUM 4 PROCESSOR

The performance goals of the Intel Pentium 4 processor compared to the Pentium III processor were 1.5X–2X higher frequency on the normal (medium) part of the chip and 3X–4X the frequency on the fast (rapid execution engine) part of the chip. These targets require aggressive domino pipelines. In the rapid execution engine, the pipeline is only eight stages deep with the last stage usually feeding the first domino stage after considerable routing. Traditional techniques such as not allowing routing into domino receivers or buffering domino inputs would have added an additional 10-20% latency to the pipe.

Accurate noise analysis using NoisePad and circuit styles such as pseudo-CMOS logic shown in Figure 8 (which provide the logic capability of domino logic and the noise robustness of CMOS) were employed.

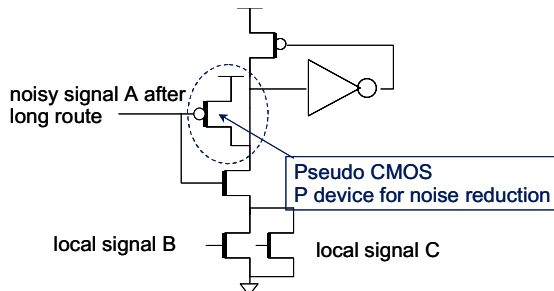


Figure 8: Pseudo CMOS circuit for input noise protection

Pulsed clocking was used in the Pentium 4 processor for lower clock power and load. This made charge sharing

protection rather difficult. To reduce power and area, dynamic latches were used extensively as mindelay blockers. These pulsed circuits have no keepers; therefore, increased noise sensitivity and charge leakage had to be verified by noise tools.

A new form of latch called the set-dominant latch was used in the Pentium 4 processor for speed optimization. This weakly held circuit node could get routed into a domino receiver causing increased noise sensitivity.

Process Optimization Consideration for Noise and Leakage

Most design rules and circuit decisions for the Pentium 4 processor, were based on early 0.18 um process specs. We wanted a robust part, which could be pushed for speed later. We expected that the transistor length and leakage targets would be aggressively pushed in our quest for speed in a mature process. Due to these considerations, we employed very large Ioff numbers for our design rules and CAD tools. As shown in Figure 9 by the process trend over time, this was indeed a wise choice: the Pentium 4 processor has scaled well in frequency and still has considerable frequency headroom speed.

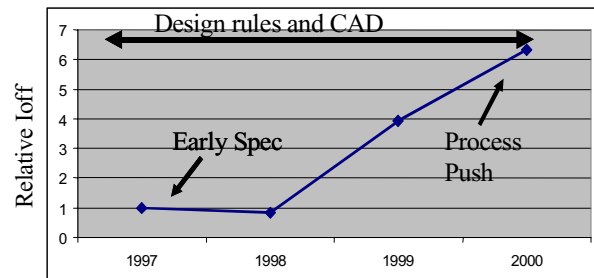


Figure 9: Impact of process push on subthreshold leakage

NOISE ANALYSIS ALGORITHMS

Some amount of noise is unavoidable in digital circuits. The question is deciding when it causes functional failure.

Strongly held static nodes recover after a noise transient and usually incur only a frequency slowdown. Dynamic latches and domino nodes, however, show true functional failure. The node goes to the wrong logic state and may not recover even after the frequency has been slowed down. Latches and other circuits with feedback show a similar failure mechanism.

Small Signal Unity Gain

Prior to our work on the Pentium 4 processor, traditional analysis of noise margins relied on the small signal unity gain failure criteria.

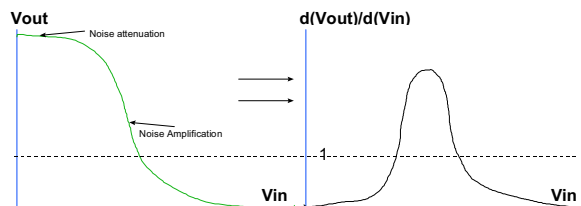


Figure 10: DC transfer function of an inverter illustrating small signal unity gain

As illustrated in Figure 10, for a small change in input noise to a circuit biased at an operating point, the resultant change in output noise is measured. If $|d(\text{Output})/d(\text{Input})| > 1$ then the circuit is considered unstable.

Unity gain is a good design metric but is neither necessary nor sufficient for noise immunity. Most aggressively designed paths have some noise-sensitive stages interspersed with quiet stages. We need to allow some noise amplification in the sensitive stage knowing that the quiet stages will finally attenuate it.

Failure Criteria: Noise Propagation

As was mentioned in the previous section, failure criteria based on unity gain tend to be extremely conservative in most cases and are still not proven to be conservative in all cases. Alternately, the entire circuit can be broken into circuit stages, across which noise propagation can be tracked. To do this, we perform an AC circuit simulation of each circuit stage, with noise sources injected in worst-case temporal fashion, combined with noise propagated from previous stages, and measure if any circuit stage failed as a result. In this case, noise can be made to propagate across any number of stages, eliminating the need for any unity gain budgeting. Failure is observed at weakly held nodes such as domino nodes and latches, where the node does not recover after sufficient time. This is very similar to path-based static timing analysis, which allows time borrowing. The computational complexity and memory cost of this approach is the main issue. We made significant CAD innovations to reduce the computational complexity of this approach and implemented this for the Pentium 4 processor in the form of a new noise simulator called NoisePad.

Combination of Noise Sources

Traditionally, different noise sources such as charge sharing, coupling, etc., were characterized separately, and individual maximum budgets were allocated for each source. This is rather conservative. A wide D2 domino NOR node, for example, is very sensitive to coupling at its inputs but has no charge sharing. Some ad hoc approaches to combining noise budgets exist, but the desirable solution is to simulate all noise sources together with no accounting for individual budgets. The simplest way to achieve this is linear superposition. The biggest nonlinear effect is the finite threshold of transistors. For example, a combination of ground bounce and coupling at the input of a transistor leads to a much larger transistor current than does an addition of currents resulting from separate ground bounce and coupling. Another nonlinearity is transistor resistance as a function of drain-source voltage. For example, the peak noise in the event of two simultaneous couplers on a line is larger than the sum of these two events, because the couplee driver resistance increases with an increase in noise magnitude. A third nonlinearity is caused by voltage-dependent parasitics. These are important, for example, when combining charge sharing with coupling effects.

Simultaneous Noise on Multiple Inputs

For multifanin circuits we have to consider not only different noise phenomena, but also their simultaneous occurrence on different parallel inputs. Traditionally, the injection of the same noise on all parallel paths was the worst-case scenario. There are several important cases such as register file arrays where this pessimism can be the deciding factor in the feasibility of the circuit. For example, in a multi-ported register file with a segmented bitline, maximum coupling cannot simultaneously occur on multiple word lines on the same port. Some background noise such as power supply noise may still be present on the other inputs.

DC vs. AC Noise Analysis

Some components of noise such as charge leakage and the low-frequency components of power supply noise have time constants much larger than those of most digital circuits. Effectively, these can be treated as DC waveforms. DC analysis and library characterization are relatively straightforward. Further, it is easy to combine noise sources; e.g., two couplers or coupling with charge sharing, with a DC approach as no computationally costly temporal shifting is required. However, noise sources such as interconnect coupling, charge sharing, etc., have pulsewidths of the same order as those required to charge or discharge most circuits. In this case, approximation of the true waveform with its peak amplitude DC produces

gross conservatism. Digital circuits work as “low pass filters” for noise due to their finite transistor resistances and load capacitances. In many matched high-speed circuits, this approximation can lead to a 2X difference in tolerable noise levels. In spite of the severe computational overhead, AC waveform analysis is necessary for the design/verification of sensitive high-speed circuits.

NOISE ROBUST CELL LIBRARY DESIGN

Traditionally, our chips have been designed with fixed cell sizes. The ability to drive different loads has been achieved by providing a finite number of different sizes and in some cases of different P/N skew. For the Pentium 4 processor, we found that additional performance, and area and power optimization, were possible by having a stretchable cell library that didn't have the constraints of fixed cell sizes. Noise robustness was an important consideration for sequential and domino cells. A key innovation for noise robustness was the use of stretchable keepers for domino nodes and sequentials. Traditionally, when assembling domino libraries, keepers were designed to keep additional delay within tolerable limits. For the Pentium 4 processor, instead of the size of keepers being hard-coded, each cell had symbolic constraints describing its leakage and noise metric (no. of pull downs, stacking, etc.), along with its delay metric. The default keeper tried to maximize noise immunity while keeping tolerable delay. As an example, wide fanin domino NORs were provided with significantly larger keepers. Similarly, stacked configurations had larger keepers. However, a designer using NoisePad, optimizing for the actual instance-based noise and speed requirements, could easily adjust this keeper strength. This did not involve creating a new custom cell (unlike other chips) and was widely used for noise suppression.

Each cell could be tuned for its noise environment (as needed) and did not have to follow conservative rules. The symbolic constraints also made the task of process conversion trivial instead of significant since the entire library did not have to be redesigned when leakage changed from a 0.18 to a 0.13 um process.

Another key decision made regarding the cell library was forecasting the optimum leakage of future processes. We predicted that leakage would get much higher for optimized 0.18 and 0.13 um technologies and therefore designed the library to combat this increase. Specifically, for the design of wide domino nodes and array and register file structures, we went with segmented bit-line architecture and disallowed circuits with large numbers of parallel pull downs (except PLA waivers). This design rule allowed us to tolerate significantly higher leakage in

the process, which is necessary for transistor performance.

Noise CAD Tool Requirements for the Pentium 4 Processor

In the Pentium 4 processor, we treat charge leakage as DC noise. Interconnect coupling, charge sharing, and noise propagation need to be handled with AC waveform analysis. All noise sources are simulated together without linear superposition. The analysis does not assume maximum budgets on individual noise sources. Regarding simultaneous noise on multiple inputs, by default the same noise is applied to all parallel paths. This can be overridden for speed or area critical paths; in which case, transient noise is analyzed on specified paths with background power supply noise on other paths.

The Pentium 4 processor is primarily custom designed with a library of parameterized/stretchable cells. In past methodologies, custom design resulted in a large overhead for noise analysis because of required characterization. In the Pentium 4 methodology, all cells are treated as custom cells with “on the fly” analysis. This requires no library pre-characterization and thus places no extra overhead on custom design.

NOISEPAD: NOISE CAD TOOLS AND METHODOLOGY

Using the technique of noise propagation, any path can be broken into small circuit stages, which can be analyzed sequentially. Technically, we could perform this analysis with industry-standard SPICE-type simulators. Unfortunately, the throughput available in the Pentium 4 processor design timeframe was not acceptable for either interactive design or batch mode verification. A new transistor-level simulator was developed that allowed a throughput that was orders of magnitude higher than the traditional SPICE approach. The key innovations were symbolic circuit simulation and simplified noise analysis of distributed interconnect.

Symbolic Circuit Template Simulation

To achieve high throughput, the noise simulator reduces/matches circuits to a list of predefined parameterized circuit templates. The differential equations governing these circuit templates have been solved symbolically in a piecewise linear manner and don't need to be solved at runtime. The simulation consists of evaluating these piecewise linear analytical solutions at succeeding time points. Device nonlinearities and voltage-dependent parasitics are dealt with because the model is “piecewise linear” and not just linear. Circuit relaxation is used for DC bias point calculations

to handle the DC noise sources. Templates exist for drivers and receivers of CMOS, domino, pass gate, and novel logic types.

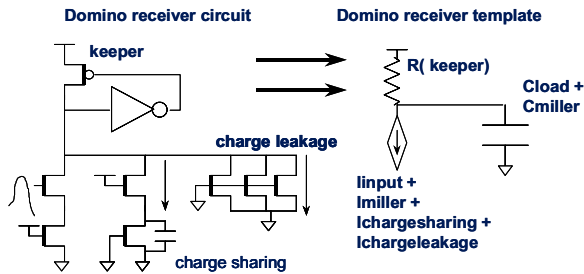


Figure 11: Circuit template idea for a domino receiver

In Figure 11, a piecewise linear waveform of input noise added to the power supply noise creates a piecewise linear current in the receiver. This current is added to other current sources such as charge leakage, charge sharing, and current injected through the gate/drain miller capacitance. The differential equation governing this circuit has a closed form solution, which is known a priori.

Transistor Models

For noise analysis, simple transistor models are often adequate. In this context, some transistors are normally “on”, in which case they try to keep a node in its correct logic state, e.g., a domino keeper. These are characterized by a large |VGS| and small |VDS|, meaning they operate in the linear region. Normally, “off” transistors are ones that try to upset the logic state of a node by current conduction. For small or reasonable values of noise, these are characterized by large |VDS| and small |VGS|, meaning they operate in the saturation region. Depending on the gate input noise, these can either be in the subthreshold or strong inversion region. With these simplifications, very computationally inexpensive transistor I-V models were developed and implemented with a precharacterized transistor table look-up model. We used a non-uniform grid to optimize for noise sensitive regions of operation; for example, we used much finer gridding in the subthreshold/weak inversion region.

Distributed Interconnect Noise Analysis

The computational complexity of noise analysis is often dominated by the coupling analysis of the distributed interconnect. In the past, interconnect coupling has been dealt with, in a lumped fashion, by putting all coupling capacitance at the end of a line. This produces significant

conservatism. Further, for interconnect with side branches, there are no straightforward solutions.

For handling complex interconnect networks, especially from post layout, Asymptotic Waveform Evaluation (AWE) analysis using iRICE has been integrated into our noise simulator.

Elmore Noise Model

To drastically increase the throughput of distributed interconnect noise analysis, a new analytical closed form approximation has been developed for multiple aggressor coupling on a distributed network.

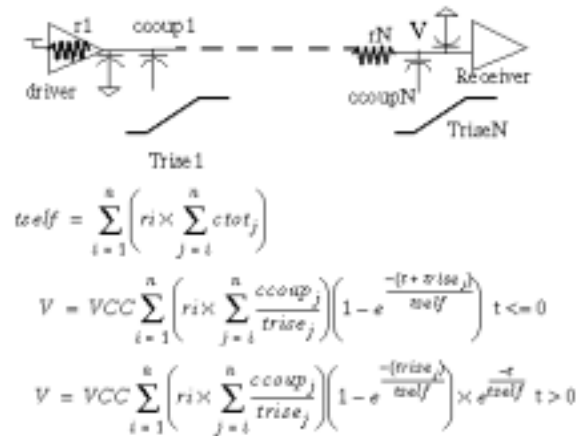


Figure 12: Elmore approximation for noise analysis

This is called the “Elmore model” due to the analogy with Elmore delay used in timing analysis. The idea here is to make the analysis much simpler by reducing the network moments or, in other words, finding the dominant time constant of the network. In Figure 12, *ctotj* is the sum of the total switching and non-switching capacitance on the *j*th node. All couplers are aligned for worst-case temporal shifts, and they finish switching at time *t* = 0. NoisePad analysis switches between this simple model and more expensive AWE models, based on heuristics.

FULL-CHIP WIRE NOISE VERIFICATION

The key idea behind the Pentium 4 processor full-chip noise verification is “strobed signaling.” A non-restoring node for noise is defined as a node, which if falsely tripped due to noise, will not recover with the passage of time (e.g., domino node or off pass gate latch). A signal is called “strobed,” if its logic cone leading to a non-restoring noise node is controlled with a clock (e.g., D1k domino). In this case, the effect of noise on this node may be dependent on clock frequency.

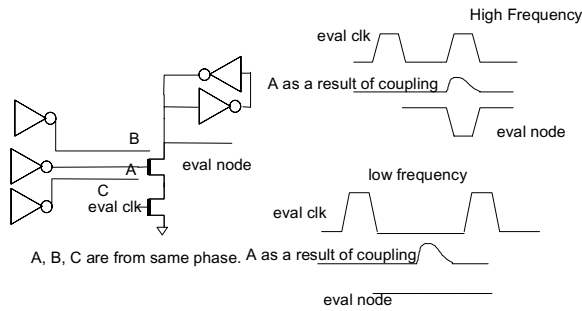


Figure 13: Impact of frequency on noise failure

As shown with the D1-k example in Figure 13, at a lower frequency, the noise will settle down before the signal is sampled and as such will not fail at the lower frequency. In most cases, the timing of aggressors switching for noise is earlier than predicted by max delay timing analysis due to a reduced Miller Coupling Factor (MCF) in the noise case. Further, the worst noise case is usually on fast silicon at high voltage (good for speed). As such, in most cases, we can ignore the cases leading to a slight frequency slowdown in our analysis. The tricky situations are those that lead to excessive frequency slowdown or even worse, frequency shmoo holes. Before spending valuable CAD tool resources on these non-trivial cases, we needed to convince ourselves that the common benign case is indeed the dominant one and therefore the one on which to base our full-chip wiring methodology.

Most full-chip signals are busses (~59,000 out of 72,000 nets), and less than 10% of full-chip signals are “sensitive” (feeding domino receivers or direct pass gate, etc.). Most busses have similar timing among different bits, which should ease the frequency slowdown and shmoo problem. Figure 14 shows the significant effect of this analysis. Most of the effect of this filtering was due to the “required filtering” that characterized frequency slowdown, and very little was due to “valid filtering,” which looks for aggressors not switching together.

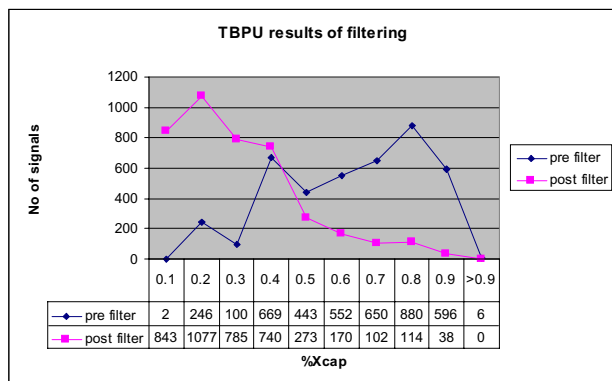


Figure 14: Impact of frequency independent timing filtering

Frequency Independent Filtering

To solve the rare cases of real noise problems on a strobed signal, we decided to classify noise issues as follows: 1) functional failure at all frequencies; 2) slight slowdown; 3) large slowdown; 4) frequency shmoo hole at a lower frequency as shown in Figure 15; 5) mindelay switching induced noise failure; and 6) excessive coupling causing gate oxide wearout. Issue number 6 was achieved simply through a VCC/2 coupling noise clamp, which was used as a warning. For the rest, we had to implement timing filtering, which understood changing timing relations at different frequencies. Timing filtering was first implemented for the Intel® Pentium® Pro processor as the tool Crosswind [4], and it introduced the concept of valid and required time window filtering; valid window noise ‘profiling’ or juxtaposition of aggressor noise over the clock period; and rudimentary modeling of drive ratios with fixed thresholds for noise sensitization. Later implementations developed for the Pentium® II and Pentium III processors improved on several aspects of driver and interconnect modeling.

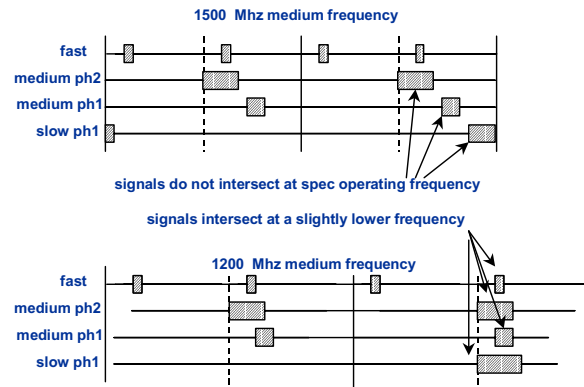


Figure 15: Frequency shmoo hole

The novel features of timing filtering for the Pentium 4 processor include three modes of frequency analysis (low frequency for burn-in analysis, high frequency for at-frequency noise and delay tests, and all-frequency sweep for noise effects); timing skew between victim and aggressors; required-time filtering with victim recovery; and an interactive graphical waveform interface for timing filter debug.

The design of the Pentium 4 processor brought new challenges to timing filtering because of the complexity of its clocking system. In earlier clocking styles, an excessive slowdown or shmoo hole was usually caused by a very late signal coupling into a signal with early-required time or by the interaction between signals from

opposite phases. In the Pentium 4 processor, however, the design incorporates several clocks that are multiples of each other: signals are F(ast), M(edium), and S(low) clocked signals. Not only do signals occur in different phases, but also with different periods. In addition, these differently clocked signals interact as they are not a priori restricted to different regions of the chip. Thus, mid-frequency shmoo holes are much more probable in such a design.

The new approach handles a clocking system with an arbitrary number of phases and an arbitrary number of synchronous clock frequencies by using a *Multi-Frequency Algorithm*.

At very low frequencies, signals activated by different phases are widely separated in time, so much so that they do not interact. This represents the low end of all frequencies to be considered, while the target operating frequency represents the high end. Sweeping frequencies at a small enough increment to catch waveform overlaps is prohibitive due to the complexity of the internal scan. We, therefore, needed a more adaptive algorithm. Here is the entire algorithm with an all-frequency sweep as its outer loop:

For each victim net:

1. Collect aggressor set for a given victim and skew timings appropriately.
2. Map clock edge references onto phases of an appropriate clocking system. For example, a set of aggressors with M and F rising edge references requires a two-phase system.
3. Perform a noise sweep, computing aggressor interaction sets and generating timing “filter table.”
4. Compute the next highest frequency of interaction among signals.
5. Return to step 2 until there is no more interaction among signals.

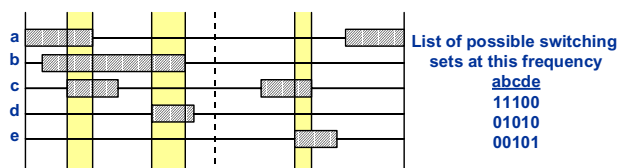


Figure 16: Illustrating logical switching set groups

The most difficult part of the algorithm is to compute the frequencies of interaction, as illustrated in Figure 16. Given that an $O(N \log N)$ scan is in the internal loop, the algorithm cannot afford to sweep with a very fine grain to catch all interactions.

The key to computing the next frequency of interaction is to comprehend the relative velocity of timing edge references as one slows the primary clock. By carefully searching the edges most close to one another and keeping track of their relative velocities, this algorithm can be made reasonably efficient. One difficulty is handling edges that refer to a previous clock phase that are actually moving backward with respect to other timing edges as frequency is increased. To handle this and other difficulties, we developed a general approach to handling both the modular nature of signal timings and measuring the frequency at which they may intersect, based on the concept of relative edge velocity.

Full-Chip Noise Convergence

Detailed noise verification requires a lot of data: circuits, timing information, detailed parasitics, interconnect, etc. For a lead processor like the Pentium 4 processor, “clean” data for all nets are available only very close to tapeout. Further, this detailed model is too slow to turn and, moreover, it is serial in nature. After finding a violation, one has to backtrack through numerous files, models, and schematics to verify if a real problem exists (needle in a haystack scenario). With these incomplete data, trending and schedule predictions are difficult.

To circumvent these problems, simple “perturbation”-based models were built using mathematical spreadsheet software. Parallel probes gather all relevant information about a net (timing, parasitics, length, circuit, etc.) to a total of 87 relevant metrics for each net! Approximately 40 full-chip models were built in one week for various “what if”(perturbation) scenarios. These models looked at tweaking various knobs: number of aggressors, switching probabilities of small aggressors, synchronization of noise propagation with coupling, probability of multiple noise events on same gate, various clock skew assumptions for timing filtering, various frequencies for allowed frequency slowdown, etc., to find reasonable settings and really serious problems but not produce too many false violations. A detailed NoisePad model was used as the starting point for these models. After this analysis, the new noise was assumed to be a slight perturbation around its NoisePad value and predicted by the change in the knob (e.g., changing lumped %xcap from 100% to 50%).

Although these fast models were very crude, they were surprisingly accurate because they did not try to predict the real noise but rather the perturbation (much smaller error). Based on these fast models, another detailed NoisePad model was built with correct knob settings and used for final convergence. As can be clearly seen from Figure 18, this exercise helped us greatly with convergence and saved us an estimated one to two

months in our noise convergence schedule. The dramatic decrease in noise violations seen in Figure 17 involved *no* work from the design team!

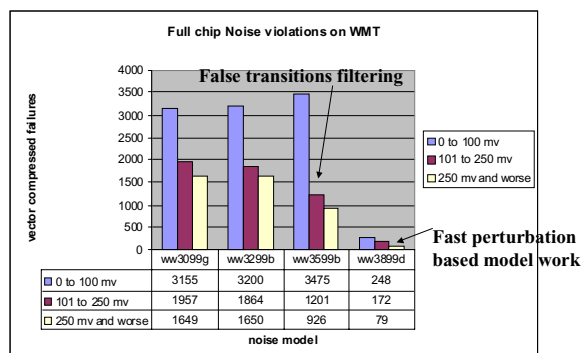


Figure 17: Road to noise convergence on the Pentium® 4 processor

MUTUAL INDUCTANCE METHODOLOGY

At low frequencies, flip-chip C4 packaging provides a very low resistance current return path. For high-speed transients, the large inductance of the package return causes significant return current to flow through the on-die power grid, as shown in Figure 18. For simultaneous switching of wide busses, the impedances in the signal and current return path can be of comparable magnitude leading to large inductive noise.

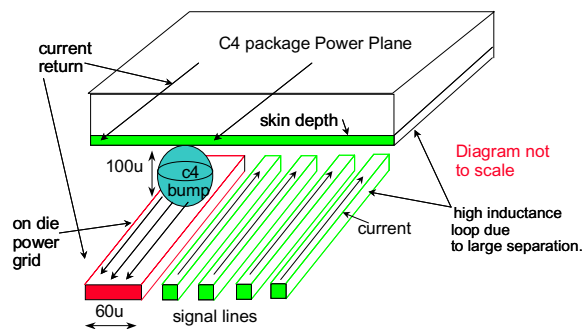


Figure 18: Signal inductance problem with flip-chip packaging

A test chip was fabricated with test structures to measure mutual inductance noise on wide busses. In this chip, signal busses of varying width could be made to switch in any combination, with several combinations of return scenarios, one of which is shown in Figure 19. We were also able to measure simultaneous capacitive and inductive noise, which helped us develop empirical design rules. To keep the area impact small while

reducing inductance, a scheme of distributed power supply was chosen for the Pentium 4 processor, where for top-level metals (M6 and M5), a power signal was routed after every 5 signal wires, thus providing a nearby current return and reducing the loop area for inductance. Towards tapeout, a tool for crude inductance estimation was written. This looked for any sensitive circuits (e.g., domino) routed for appreciable distance in the neighborhood and parallel to long, wide busses. By taking the width of the bus, distance from the bus, and length of overlap, an inductance noise metric was used to flag any possible problems. This check was not restricted to wires routed in the same metal layer.

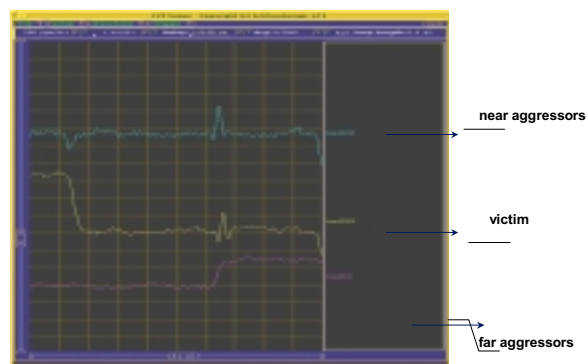


Figure 19: Silicon measurements showing inductive noise

TIMING AND NOISE INTEGRATION

Traditionally, timing analysis (PV) has remained decoupled from noise analysis. As we push both timing and noise limits, there is increasing interaction between the two.

Currently, min delay analysis verifies that all circuits meet their hold time limits while a pulse width/delay check verifies that pulses are wide enough for circuits. In the 0.18 um technology generation, the tool Pathmill* is used for min delay analysis. The common algorithm for hold time checks is to ensure the switching data signal does not reach its 50% point before the going away clock reaches its 50% point.

*Other brands and names are the property of their respective owners.

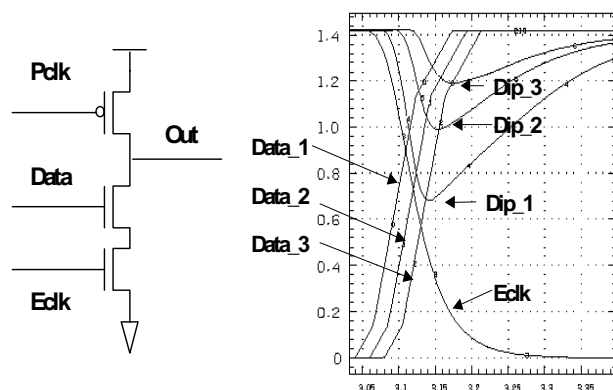


Figure 20: Timing-induced noise

There are some other algorithms, which change the threshold (50% point) or move the check to data output rather than input. These algorithms are inherently flawed because they do not take into account the context-dependent noise robustness of the circuit.

In Figure 20, taking any of the measured values as hold time for a circuit would be completely arbitrary if you didn't know the circuit's noise margin and the other sources of noise that were present. A pulse width/delay checks that the pulse to a circuit is wide enough for it to reach within a certain voltage of a full transition. This check is again arbitrary, without knowing how sensitive that circuit is to incomplete transitions (noise). As an example, we found that a default mindelay Pathmill analysis of the Pentium 4 processor domino library showed several instances where a Dlk circuit passing default mindelay (hold checks) would leave a glitch at the domino output that was large enough to cause a complete false transition after the high-skewed static stage. Currently, no design flow would catch these problems, thus causing potential silicon bugs.

Our response was to treat hold checks and pulse width checks as an analog glitch check. The glitch amplitude corresponding to a certain hold time is automatically injected into the noise tools and propagated to succeeding stages to ensure circuit functionality. Thus, we can make tradeoffs between min delay and noise requirements. This new source of noise is combined intelligently and not just added to other traditional sources of noise, such as coupling, taking into account events that are possible logically at the same time. This tradeoff was used quite widely for critical circuits.

Since the design of the Pentium 4 processor, all Intel® timing characterization tools take simultaneous noise margins into account when doing timing analysis for hold, set up, and pulsewidth checks.

SUMMARY

Key findings from the Pentium 4 processor noise and wire design methodologies and CAD tools have been presented. By a combination of aggressive circuit design, short, high-density wiring and noise methodology, and the appropriate CAD tools to help design and verify these, the Intel Pentium 4 processor looks poised to be a successful, fast, reasonably small die product. We have shown that an architecturally larger chip need not lead to longer physical wires if careful methodology and repeater design are used, thus enabling higher frequency. Very aggressive circuit styles have been allowed by innovations in noise CAD tools, which will enable even higher frequencies. High density has been enabled by improved noise methodology, thus allowing aggressive, dense wiring with judicious use of spacing and shielding. The inductance problem, although significant, has been accounted for in the design by our distributed power grid. Circuit styles and a methodology that are robust for leakage will allow us to push the process for speed. Tradeoffs between timing and noise have been enabled by innovations in CAD tools. In general, a lot of care and effort has been put into noise immunity to create a chip that should work robustly in the field. Much of this methodology and CAD tool ideas can be incorporated into future chip designs.

ACKNOWLEDGMENTS

We acknowledge all present and past members of our noise group for the all-nighters and Brad Hoyt for discussions. We also acknowledge our DT co-development partners. We acknowledge Paul Madland for guidance and for having a feel for where the silicon problems would really be. And, we acknowledge our management for sticking with our full-chip wire/noise direction, even though at the time, it looked quite risky.

REFERENCES

1. Rajesh Kumar, Eitan Zahavi, Desmond Kirkpatrick, "Accurate design and analysis of Noise Immunity for high-performance circuit design," *Design and Test Technology Conference (DTTC) 1997*. Intel internal document.
2. Eitan Zahavi, Rajesh Kumar et. al., "Novel Methodology and Tools for Noise Immunity Design and Verification," *DTTC 1998*. Intel internal document.
3. Madhu Swarna et. al., "Integrated timing and noise characterization of sequentials for accuracy and increased design space," *DTTC 2000*. Intel internal document.

4. Conley, Kirkpatrick et. al., *DTTC 1995*. Intel internal document.

AUTHOR'S BIOGRAPHY

Rajesh Kumar is currently a Principal Engineer in the Desktop Platforms Group. He received an MSEE degree from the California Institute of Technology (CalTech) and a BTech degree in EE from the Indian Institute of Technology. He joined Intel in 1992 as a designer of the X86 Instruction Decoder of the Pentium Pro processor, working in the areas of microarchitecture, logic, circuit design, and silicon debug. He did the initial research on fundamental circuit limits to high-frequency pipelining, enabling the rapid execution engine for the Pentium® 4 processor. He led the methodology and CAD work for noise, inductance, interconnect, leakage etc., for the Pentium 4 processor. He was the founder and initial chair of Intel's taskforce on crosscapacitance. His current interests are in high-speed/low-power design, parallel/DSP computing architectures, novel non MOSFET devices and conscious computers. His e-mail is rajesh.kumar@intel.com.

Copyright © Intel Corporation 2001. This publication was downloaded from <http://developer.intel.com/>.

Legal notices at <http://developer.intel.com/sites/developer/tradmarx.htm>