DesignCon 2008

Counting the Picoseconds: Integrating Timing, Signal and Power Integrity Analysis

Doug Burns, SiSoft
dburns@sisoft.com 978-461-0449

Todd Westerhoff, SiSoft
twesterh@sisoft.com 978-461-0449

Jeff Boyd, Denali
jboyd@denali.com

Abstract

High-speed parallel interfaces now operate at speeds in excess of 1600 MT/s, with timing margins well under 100ps. At these speeds, detailed timing / SI analysis is needed to ensure adequate voltage and timing margins. Power integrity analysis gets added to this mix, predicting how switching noise will affect the interface's timing budget.

Device timing is measured and guaranteed under specific conditions that drive the details of how signal/power integrity simulations must be performed and simulation results measured. Understanding these details is essential to combined timing/SI/PI analysis that correctly predicts interface operating margin.

Author Biographies

Doug Burns is SiSoft's Vice President of consulting services group and has over 20 years experience leading teams in ASIC/system hardware design and signal integrity. Doug holds seven patents. Doug's expertise spans a wide range of engineering disciplines including: high-speed system design, system architecture, VLSI design, package design, interconnect analysis, crosstalk analysis, electro-magnetic modeling, I/O buffer analysis, SSO analysis, clock distribution and skew analysis. Doug graduated Magna cum Laude from the University of Massachusetts with a BSEE degree and received an MSEE degree from Northeastern University.

Todd Westerhoff is SiSoft's Vice President of software products and has 28 years experience in modeling and analysis of electronic systems. Prior to joining SiSoft, Todd managed a high-speed design group that provided static timing, signal integrity and design rule consultation to various engineering groups within Cisco. Todd holds a B.E. degree in electrical engineering from the Stevens Institute of Technology in Hoboken, New Jersey.

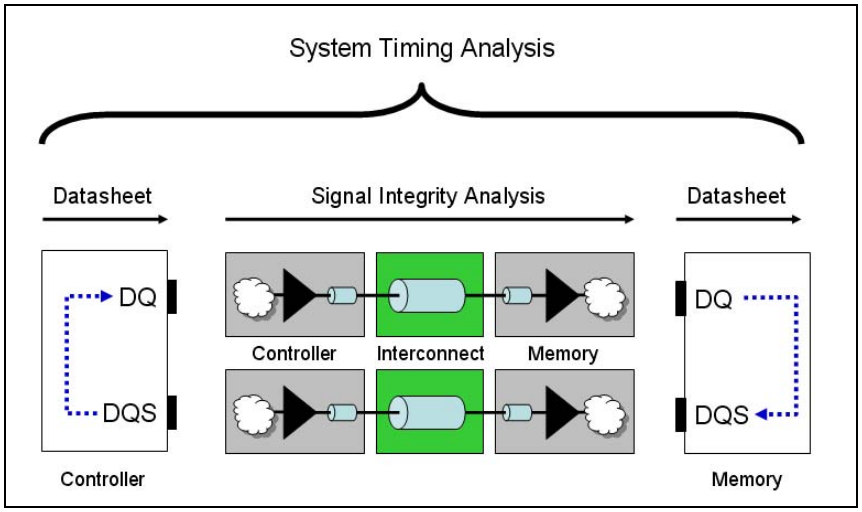
Jeff Boyd is a Staff Design Engineer with Denali Software and works in the Databahn Engineering group specializing in signal integrity and system solutions. He has 22 years experience in high-speed Systems and ASIC design. Prior to joining Denali, he has worked in product development for Tandem Computers, Hewlett Packard and Conexant Systems. He holds a BSEE from the University of Illinois.

High-Speed Parallel Interface Design

Problem Statement

High speed parallel interfaces now operate at speeds in excess of 1600 MT/s, which just a few short years ago was the domain of serial links. To achieve these high speeds, detailed timing and signal integrity (SI) analysis needs to be performed, and the effects of crosstalk and power integrity (PI) analysis must be included to ensure designs will function reliably.

Interfaces must be analyzed from a system perspective, combining component timing specifications, interface operating speeds and interconnect delays together to determine timing margins. Signal integrity simulations perform detailed analysis of digital I/O switching behavior and provide interconnect delays, or "flight times" that are plugged back into the interface timing budget to determine operating timing margins.



While it is easy to state what needs to be done at an abstract level, the key to a successful implementation is having a methodology that closes the loop between signal integrity, power integrity, and timing analyses. This methodology must encompass the SI and timing effects of the source device, packaging, system interconnect, power noise, crosstalk, and the receiver device. In the following discussion we shall explore the design of an ASIC DDR3 memory interface, outlining the key facets of a successful design process.

Getting Started

Analysis of a high speed parallel interface should start with a carefully crafted timing budget, accounting for the different physical effects that are expected to be significant. This initial budgeting helps prove design feasibility and establishes elements of the timing budget that can be influenced by the system implementation (thereby identifying critical aspects of the design).

Calculating the setup and hold margin for a data transaction requires understanding the detailed min/max delay values for each component. Since detailed information is often unavailable at the beginning of a project, an alternative approach is to assess the data uncertainty window (max-min) for each timing budget element, adding items up to see if the total uncertainty is greater than the available timing window.

Note that multiple timing budgets are generally required to account for all the timing requirements of a particular interface. For our DDR3 controller we will need to keep track of:

1. Address/command to CLK relationship @ DRAM
2. Control to CLK relationship @ DRAM
3. DQS to CLK relationship @ DRAM
4. DQ to DQS relationship @ DRAM (write operations)
5. DQ to DQS relationship @ controller (read operations)

We separated address/command from control in the above list because the connectivity (and therefore signal loading) can vary between these signal groups, depending on the memory configuration employed. The differences in simulated interconnect delay can be large enough that timing for each signal class should be considered separately.

The "fly-by" routing strategy used for DDR3 DIMM clock/address/command/control signals causes these signals to arrive at each DRAM staggered in time. Since a DQS to CLK relationship needs to be maintained at each DRAM, this necessitates evaluating clock/address/command/control/data/strobe relationships on a per-lane basis. This increases the complexity of creating and managing all the timing budgets needed to design and validate the interface.

For simplicity's sake, we'll pick one of the timing budgets and follow it through the design and analysis process. Table 1 shows an initial timing budget for a single lane DDR3 write transaction using the "uncertainty window" method.

Skew Component	Mnemonic	Uncertainty	Units
ASIC Clock Distribution (Data Skew)	Tdq2dqs	20	ps
ASIC Clock Jitter	Tdqs_jit	20	ps
ASIC Clock Duty Cycle (5% CK)	Tdqs_dcd	80	ps
ASIC DLL Jitter	Tdll_jit	35	ps
ASIC DLL Phase Error	Tdll_pe	60	ps
ASIC IO data asymmetry	Tdq_rf	40	ps
ASIC SSO	Ta_sso	100	ps
ASIC Package Routing Skew	Tpkg_skew	25	ps
PCB Interconnect Routing Skew	Tpcb_skew	20	ps
PCB Interconnect ISI	Tpcb_isi	100	ps
PCB Interconnect Crosstalk Skew	Tpcb_crosstalk	50	ps
Dram Setup	Tr_setup	125	ps
Dram Hold	Tr_hold	125	ps
Total Uncertainty		800	ps
Bit time at 1600MT/s		625	ps
Worst Case Timing Margin		-175	ps

Table 1 Sample uncertainty timing budget for DDR3 DQ write operation

Keeping it all in perspective

In this example, we have assumed a custom ASIC and itemized the full system timing budget from the clock source (in the ASIC) to the clock destination (at the DRAM). Each skew component represents a piece of the data path from the ASIC to the DRAM. Items have been included in the budget based on an understanding of the detailed implementation and where critical timing points are. It's always tempting to include as much detail as possible, but applying too much detail can become counterproductive. For example, consider the delay skew imposed by a chain of 6 inverters, each with a rise delay of 25ps and a fall delay of 20ps. The delay uncertainty imposed by each inverter is 5ps, so one could conclude the chain will contribute 30ps of timing uncertainty to the circuit. In reality, the inverter chain would contribute no timing uncertainty it includes an even number of inverters with identical characteristics. Building a good timing budget requires understanding how the circuit will actually behave – simulators are not magic, and there's no substitute for knowledge.

Timing budget driven analysis

Our initial timing budget in Table 1 shows a timing failure of 175ps. It's common for an initial budget to show negative or small positive margin (especially at these speeds). In

cases where the initial budget shows a large positive margin, minimal design work may be required – a design using "rules of thumb" may suffice, using only basic SI analyses to look for signal quality. When an initial timing budget shows large negative margins, one needs consider carefully which parameters can be affected by design implementation, and which parameters should be considered as fixed. If a large enough portion of the budget is fixed, a fundamental re-architecting of the interface or operation at a slower speed may be required. No matter what margin your initial budget shows, the budget can drive design and analysis decisions, identifying what issues need to be addressed to ensure the design can operate reliably. Looking back at Table 1, we note that SSO and ISI account for 200ps of uncertainty, the ASIC clock system accounts for 120ps, the DLL accounts for 95ps, and the DRAM accounts for 250ps. By focusing on the largest contributors to uncertainty first, we make the design and analysis process as efficient as possible.

Signal Integrity Analysis Plan

Once critical areas of the design have been identified, the next step is to establish an overall simulation strategy. This means defining what simulations are required, what simulation models are needed, which elements of the timing budget should be included in the same simulation, and how simulated behavior should be measured for inclusion in the timing budget.

It might be counterintuitive, but one of the goals of a well-crafted signal integrity analysis plan is defining the minimum modeling and simulation needed to prove that the design will work. Accurate modeling and simulation require considerable effort, expertise and money, none of which should be expended without knowing exactly what level of detail is required, and why.

What needs to be simulated

When looking at a parallel interface and the number of detailed physical effects that need to be assessed, it's tempting to model as much of the problem as possible in the same simulation. The use of one large, coupled circuit allows multiple signals (address, clocks, data, control) to be modeled, any type of data pattern to be run and the effects of crosstalk, SSO, ISI, topology, etc. … to be explored. While this provides the ultimate in flexibility, it also requires the most complicated model. It's common for a fully-coupled simulation to reflect 36 or more signals. Building interconnect (electromagnetic) models for complex coupled simulations can require weeks of effort, and the resulting simulations are prone to convergence issues. Fully coupled simulations require additional memory and compute resources, with simulation runs sometimes taking days to complete.

An alternative approach segments the problem into multiple independent transactions (DQ vs. DQ, DQ vs. DQS, DQ vs. Address, etc) and the simulations needed to support them. Smaller simulations require simpler interconnect models, are quicker to set up and run and tend to be more stable from a simulation standpoint. The risk with this approach is that some of the coupling between signals can get overlooked, if it's not modeled using a targeted simulation and not accounted for elsewhere in the timing budget.

The transactions that must be analyzed for our DDR3 interface include:

- Address vs. CLK
- Address vs. Address
- Address vs. DQ (Read and Write, DQ victim and Address Victim)
- DQ vs. DQS (Read and Write)
- DQ vs. DQ (Read, Write, Read followed by write, Write followed by read)

For each transaction, we need to define what physical effects need to be modeled and simulated. Interactions (coupling) between physical effects and timing margin drive the details of what simulations get performed. While signal integrity analysis can range from simple to complex, the key output is an interconnect delay or skew value that can be brought back to the system timing budget.

What must be simulated together

Whenever a design problem is broken into pieces, assumptions must be made about each of the pieces, how results are measured and how the results are to be combined. When the results of independent analyses are combined, "integration error" occurs when different analyses use different assumptions or when a common parameter (such as voltage) is included in each simulation and the combination of the simulation data "double counts" the effect. The designer must understand the interactions between circuit elements as they relate to the budget items and the analyses behind them.

As an example, let's look at the analysis of the DQ bits. We know that we will see SSO effects when all bits switch together and that these effects will be based upon the load, switching frequency, and power distribution network. We know that ISI will be a function of the interconnect length, switching frequency, driver impedance state, and data pattern. We also know that crosstalk will be a function of the etch/via length and spacing, switching rate and edge rate. How should we structure simulations to capture these effects and bring them back to the timing budget?

Let's first consider the case when we try to run separate analyses for SSO, ISI and crosstalk.

We could capture SSO's effect on timing by modeling multiple signals connected to a coupled power network model. The signal interconnect could be uncoupled as we're trying to isolate timing shifts due to SSO effects. We would excite the network by driving all 1's followed by all 0's, and compare that to a nominal case with only a single output switching. ISI effects on the signals would be present in our coupled network, although the timing shifts due to ISI would be canceled out (theoretically) because ISI would be present in both simulations. Standalone ISI simulations could be run in a second simulation with an uncoupled network with ideal power, using complex stimulus pattern to excite the worst case ISI. Crosstalk patterns could be run in a third simulation with ideal power, coupled interconnect and simplified data patterns.

Each of these simulations would present an uncertainty that could be summed up in the overall timing budget (look at **Table 1** again and note that we budgeted a skew component for each of these effects).

Is this a suitable approach for this problem? A review of the physics behind the 3 effects (SSO, ISI, and crosstalk) makes it clear that ISI will be affected by the output buffer drive impedance, which is a function of the voltage on the driver power rail, which is in turn a function of the power network and previous switching behavior (SSO). Crosstalk is also affected by the driver's power rail, and will in turn affect ISI. If we explore the situation in detail, we'll probably find that the worst case SSO occurs with a different data pattern than the worst case ISI pattern. These interdependencies mean that there is a good chance of overestimating the combined effect on interface timing if we break these simulations up.

Creating a full coupled network simulation to evaluate SSO, ISI and crosstalk simultaneously will provide a better assessment of the design's timing margins, but also represents a considerable model development and validation task. Here again, the timing budget guides us as to whether it's worth the effort. The legitimate need for a detailed coupled model occurs when electrical behaviors interact and affect the interface's overall timing or voltage margins to the point where it becomes the difference between success and failure.

Model Development

Once the required simulations have been identified, we need to create models to support those simulations. These are more than just SPICE models. We need to capture detailed timing and electrical signaling levels from the device datasheet, obtain and validate IO buffer models, then build electrical interconnect models for the IO, packages, and PCB interconnect.

We already noted that SSO, ISI, and crosstalk uncertainties account for a significant part of our example design's timing budget. We could run three analyses, one for the SSO, another for ISI, and a third for crosstalk, but the interaction between these elements would be lost. Since our initial timing budget is significantly negative (175ps / 625ps = 28% of the cycle), the best strategy for our DQ-Write case is probably a fully coupled analysis. To perform this simulation, we will need the following models:

- HSPICE model of the ASIC IO (IBIS models will not suffice since this simulation utilizes the power rails)
- ASIC package model including IO power delivery (2D or 3D model)
- On-die ASIC decoupling model per IO
- Package decoupling model (if any)
- PCB etch models (lossy wline, coupled) (2D)
- PCB Via models (3D)
- PCB decoupling model
- DRAM package model

- DRAM receiver model
- Voltage measurement levels (based on component timing specs)
- Timing metrics
- Standard load model

Model Accuracy

Taking the time to validate each model is a key part of the methodology. Without proper models and measurement methods, simulation can become an open-ended process that consumes lots of time and money, but doesn't really reflect how the design works. Models can be created at varying levels of detail, and the simulators that use them are always based on specific assumptions. For example, an output buffer can be modeled as a behavioral element that uses tables to define its behavior (IBIS), or as an interconnection of transistors and associated parasitic values that are derived from the circuit's physical layout.

We tend to make the assumption that a more detailed model provides a more accurate representation of real-world behavior, and is therefore automatically better. This isn't necessarily true. We tend to overlook the fact that more detailed models take longer to run, more effort to create and more effort to validate. To be successful, a designer needs to understand each model used, knowing what behaviors it represents and what behaviors it doesn't. That knowledge should be incorporated into how simulations are set up and reconciled against the timing budget. This is time-consuming work, and it's tempting to "err on the safe side" by opting for the most detailed models available, assuming they model all the needed effects for a given analysis. The problem with this approach is that if you don't have the time to properly sort out your modeling strategy (opting for "detailed" models as a fail-safe bet), then you probably don't have the time to validate your models or analysis results either.

Relying on models that are assumed to be "accurate" (but not validated) can lead the user to a false sense of security. Simulators aren't magic – they're computer programs, and are subject to all the same limits as any other piece of computer software. They're great tools when used properly, but will rarely give the right answer in the absence of experience or good engineering judgment.

How detailed do models need to be? They need to be only as accurate as required to model the effects being studied in a particular simulation. When is a model "accurate" enough? When the results of simulation match measurement to a reasonable degree (typically 5%) for the effects being studied, we tend to consider that model accurate. In a well tuned design process, the results from a theoretical approach to the problem, computerized analysis, and physical measurement all match. We call this concurrence an "Engineering Hat Trick".

Precision vs. Accuracy

There is an important difference between accuracy and precision. Accuracy describes the degree to which a simulation model represents real-world behavior. Precision, on the other hand, describes the level of detail that can be consistently reproduced in an analysis. The problem here is that simulation results are arbitrarily precise, but not arbitrarily accurate. A simulator can be directed to report a waveform's voltage with any number of decimal points, or model waveform behavior with an arbitrarily small time step, but that doesn't necessarily make the results more accurate, it just makes them more precise.

Component Specifications

Component datasheets document the exact conditions under which device timing is guaranteed. These specifications can be complex, varying based on device pin class, operating speed, rising vs. falling edge and input slew rate. Understanding these specifications requires effort, yet is essential to ensuring delays derived from SI/power analysis can be correctly incorporated into an interface timing budget. Timing specifications identify the driven load; min/max slew rates, timing measurement points, DC min/max input voltage levels, AC min/max voltage swings, as well as how delays vary as a function of slew-rates.

Waveform Quality & Interconnect Delay

There are two types of processing performed on simulation results:

- Waveform quality analysis ensures that signals conform to key metrics at the receiver's input. Examples of quality metrics include overshoot specifications, (ensure signals won't damage a receiver's input circuits), along with non-monotonicity, ring-back, and slew-rate checks. Violations of waveform quality criteria may render extracted interconnect delays invalid, in which case those delays can't be plugged back into the system timing budget.

- Waveform timing analysis extracts interconnect delays for inclusion in the system timing budget. This is the simulated system-level delay from the driving to receiving device, where the "start time" is based on how component output timing is specified at the driver, and the "stop time" is based on how input setup and hold times are specified at the receiver. This is simple in concept but can become complex in practice. The practice of using separate "reference" simulations (to standard load) to normalize the driving device's output delay is well understood, but the same is not true at the receiving end. Input setup and hold specifications are based on specific measurement voltages and slew rates at the input pin. System conditions rarely duplicate those reference conditions, and the resulting differences in delay through input buffers needs to be included in system-level timing analysis. For DDR2 and DDR3 devices, this adjustment is made through the use of slew-rate de-rating tables, which modify the device's setup and hold requirements as a function of the edge rates of the input signals.

Unique modeling constraints

The signal integrity model and simulation strategy may need to include effects that are unique to the technology and interface being analyzed. Examples include:

Model overrides

Sometimes more than one simulation model is required for a device, or a single model may have different operating modes. This is common for devices with on-die termination (ODT), where the device has one behavior when driving, and one of several different behaviors when receiving (depending on the ODT mode selected). Simulations for a DDR3 memory device typically use a non-terminated output model when the controller is reading data and one of several different input termination models when the controller is writing data to the memory.

The simulation strategy for the interface must take this into account, or the actual device's behavior will not be modeled properly. Automating the analysis process (i.e. identifying which models to use for which cases and switching models automatically based on the simulation case) simplifies the designer's task.

Populations

"Populations" occur when a system can be configured in multiple ways. A common case is a motherboard where different memory modules can be plugged into each slot. Since each memory module may have different loading characteristics, differing signal integrity behavior impacts system timing and voltage margins. Each change to the design potentially requires rerunning a simulation regression with all the supported memory configurations – this is another process that becomes burdensome if not automated well.

Power Integrity

Power Integrity is a broad term that relates to the various issues with supplying power and ground to the different components in a system. The power regions for an IC can be separated into the IC's core and I/O ring. Models of the package power distribution and the PCB power distribution are created to allow power transients to be studied. Core power requires unique models of the internal switching behavior of the IC and is generally modeled separately from the signal integrity analysis. For purposes of I/O signal integrity simulations, core power is generally modeled as a DC source with a defined min and max value.

Component specifications drive the need for performing SSO simulations. The designer must first determine whether a device's AC specifications include SSO effects, or whether timing is specified with a single bit switching. This can be challenging as some vendors are reluctant to provide this data. When specifications don't include SSO, a budgeted SSO value based on analysis should be used.

Performing an accurate analysis of die/package level SSO is a complex task. Additional noise observed when multiple I/O's switch is generally termed SSO. There are several different contributors to this behavior:

- Signal – signal crosstalk within the package
- Crosstalk due to shared signal returns within the package
- On-die power rail collapse

Modeling these behaviors accurately in a simulation model is a complex task, requiring models for the I/O ring, die-level decoupling / parasitics and coupled model for the package interconnect and power distribution. These require large simulation runs than are prone to convergence issues. Setting up and running this level of simulation typically requires weeks worth of effort. This represents the next level of complexity, where multiple I/O and a combined power/interconnect circuit model are combined to assess the effect that shared return paths and decoupling strategies have on interface delays.

When device timing specifications exclude SSO and good modeling data isn't available, a budget number can be used. This requires good engineering judgment, backed up by some simple practical simulations. A rough model can be built based on package dimensions, number of IO and power pins, and the package technology, along with the I/O buffer model and assumptions about on-die decoupling. This is obviously not ideal, but a budget number is still better than neglecting the effect in the overall timing budget.
.

Executing to Plan

Once validated models are available, simulations can be performed and results measured. Simulations should be run within a process that allows easy manipulation of design variables and that provides output data in a concise and predicable fashion. While this is simply stated, it is a key aspect of having a working design process

Results Processing

Given all effort goes into producing simulation results, it's critical that we sift through these results in an organized fashion. Data extracted from simulation results produces the timing data fed back into our system timing budget. In our example, we ran simulations that combined the SSO/ISI/Crosstalk effects and captured rise/fall asymmetry in one simulation. From these results, we extract the min/max data delay data to provide a new uncertainty value for the timing budget (Table 2). Note that multiple items from our original budget have been "zeroed out" and that a new budget item has been added to represent their combined uncertainty in the timing budget.

Skew Component	Mnemonic	Budgeted Uncertainty	Simulated Uncertainty	Units
ASIC Clock Distribution (Data Skew)	Tdq2dqs	20		ps
ASIC Clock Jitter	Tdqs_jit	20		ps
ASIC Clock Duty Cycle (5% CK)	Tdqs_dcd	80		ps
ASIC DLL Jitter	Tdll_jit	35		ps
ASIC DLL Phase Error	Tdll_pe	60		ps
ASIC IO data asymmetry	Tdq_rf	40	0	ps
ASIC SSO	Ta_sso	100	0	ps
ASIC Package Routing Skew	Tpkg_skew	25		ps
PCB Interconnect Routing Skew	Tpcb_skew	20		ps
PCB Interconnect ISI	Tpcb_isi	100	0	ps
PCB Interconnect Crosstalk Skew	Tpcb_crosstalk	50	0	ps
Dram Setup	Tr_setup	125		ps
Dram Hold	Tr_hold	125		ps
Simulated SSO/ISI/Xtalk/assymmetry		0	150	
Total Uncertainty		800	670	ps
Bit time at 1600MT/s		625	625	ps
Worst Case Timing Margin		-175	-45	ps

Table 2 Final DDR3 Write Timing Budget

Interpreting Results

Integration Error

When interface analysis is partitioned into multiple independent simulations, it is critically important that the conditions under which each analysis is performed are controlled and understood. As an example, setup and hold constraints for devices are always provided based on measuring the input signals at specific voltages and slew rates. Since signal integrity analysis provides interconnect delays for inclusion in the system timing budget, it's important that those delays be measured in accordance with how the component timing was specified.

"Integration error" is uncertainty introduced into the interface timing budget when the results of independent analyses are combined. There are several possible sources of integration error:

- Specification error
- Summation error
- Coupling error
- Conversion error

Specification error occurs when different elements of the timing budget are specified or analyzed based on differing assumptions. Incompatibility between component timing specifications and signal integrity waveform timing measurements is a common problem.

Consider the case where the setup time for a DDR3 rising data input is specified at 575mV (VDDQ/2 – 175mV), but signal integrity interconnect delays are measured to VDDQ/2. This results in a 175mV difference between the component timing spec and the simulation delay. If the input signal to the memory slews at 2V/ns, an error of 87.5ps will be introduced due to the difference between the two voltage specifications.

Another common source of specification error is confusion about where in the circuit timing data is specified. Component timing specifications are always represented as a delay from one point on the device to another, predicated on specific measurement conditions. The details matter. If the device timing is specified to the component pin, then the signal integrity analysis simulations and measurements should take that into account. If the device timing is specified at the device pad or the interface between the chip's core and I/O ring, the signal integrity analysis should be updated accordingly, or elements of the timing budget will either be double-counted or omitted.

Summation error can occur when results from different analyses are added into the timing budget. The simplest way to combine budget items is to add them together, but some items have to be "correlated. For example, it doesn't make sense to add a worst case simulated interconnect delay to a best case timing spec – the device will either be fast or slow, but not both at the same time. Mixing delay "cases" will increase timing uncertainty and provide overly pessimistic results.

Sometimes simply adding budget elements together creates an issue. Computing a delay through a chain of elements can be overly optimistic or pessimistic when the best/worst case delays for each element are added linearly. Some people advocate using techniques like root sum squared (RSS) combination to predict the overall delay of such chains, and many ASIC design tools have specific methodologies for measuring delay variance through paths in a circuit.

The important point is that the method used to combined budget elements – linear addition, correlated addition or RSS combination – affects the computation of the overall design margin. Estimating the difference between the different methods and understanding the correlation to real world behavior is key to design success.

Coupling error occurs when physical phenomena are analyzed and budgeted separately, but interact in real world design operation to affect the design's operating margin. We've already talked about SSO/ISI/Crosstalk analysis with our DDR3 example. We can extract all the power and signal coupling between the driver and receiver, and perform one massively coupled analysis that includes as many physical effects as possible. The challenge isn't determining whether a massively coupled analysis is a better direct approximation of circuit behavior (of course it is!), but whether it's worth the time and effort.

Extracting a coupled signal / power delivery model is a task that requires considerable time, expertise and computer power, and those models must be regularly correlated against measurement to ensure the models are useful. Extraction and correlation of these models can add weeks or months of time to a project, in addition to the cost of modeling software and measurement equipment.

As with summation error, the important point about coupling error is to have an understanding of its magnitude, so that any error introduced by the analysis methodology can be weighed against the design's predicted operating margin. If the timing error introduced by separating SSO and interconnect delay simulations is estimated at 75ps, and the delay's operating margin is -25ps, then the additional time and expense of detailed model development and correlation would be justified.

Conversion error can occur when data is converted from one form to another for purposes of analysis. For example, it's common to provide a coupled interconnect model as S-parameter data. In order to perform simulation in the time domain, the simulator must convert data between the frequency and time domains, which introduces issues of its own. In addition to creating problems with simulation convergence, converting this data introduces some numerical error, which may or may not be significant in a given analysis.

The terms "specification error", "summation error", "coupling error" and "conversion error" are presented as examples of what must be considered during a budget-based analysis. Good high speed design is the practice of informed approximation – breaking the problem into appropriate pieces, analyzing each piece independently, then putting the results back together, being mindful of any error introduced in the process and the influence on the final result.

Reuse

As we've seen, performing a thorough and accurate analysis of a high speed interface requires considerable time, planning and effort. Fortunately, once analysis has been performed and the design margins have been established, much of the work can be re-used for subsequent projects. Reuse of analysis data can usually be divided into two main stages:

Pre-layout analysis – involves the ability to re-use the same simulation models, timing budgets and setups from a previous project to derive physical design rules for a new design. Ideally, users would be able to quickly modify interconnect models to reflect their new design, and reuse the rest of the data from their last analysis.

Post-layout analysis – ideally uses the same timing, I/O model and interconnect topology information created during pre-layout analysis, but replaces the "pre-layout" topologies with models extracted from the completed physical design. Since pre-route analysis typically only analyzes one "representative" net for each net class, post-route analysis also requires identifying multiple independent nets as members of the same net class, performing the appropriate analyses and making the corresponding measurements, then consolidating analysis results accordingly.

A methodology that allows a pre-route analysis setup to be directly reused between designs, and that leverages pre route setups to automate post-route analysis – is ideal.

Conclusion

As we've seen, accurately predicting timing and voltage margins on a high speed interface requires integrating the results of signal integrity and power analysis with component timing data. Component timing specifications, in turn, drive how signal and power integrity results should be measured. Defining a detailed timing budget at the start of a project helps establish feasibility, identify critical areas where detailed analysis will provide the most benefit, and guide decisions for partitioning simulations to model different physical effects.

The process we've described is an iterative one where analysis occurs in stages. A preliminary timing budget outlines design margin, identifying critical elements for detailed evaluation. The design's predicted operating margins drive the degree of detail to which behavior is analyzed, which keeps things in perspective. Proper model development, validation and simulation are both time-consuming and expensive. The experienced high speed designer creates a modeling and simulation strategy that is as detailed as the situation requires, but no more.

.