Tool Capabilities needed for Designing 100 MHz Interconnects

Tim A. Schreyer

Intel Architecture Labs
Intel Corporation
Hillsboro, OR 97124, USA
Tel: 503-264-8332
FAX: 503-264-6053
e-mail: tim_a_schreyer@ccm.intel.com

Abstract – Printed circuit board design complexity increases greatly as bus speeds exceed 100 MHz. This increased complexity is due more to the large number of simulations a designer must complete rather than simulation or modeling accuracy. This paper presents the case for these increased numbers of simulations, and presents techniques for managing this complexity.

I. INTRODUCTION

Many system buses are now or will soon be operating at speeds of 100 MHz or greater. In today's personal computer, these speeds can already be found on the CPU's system bus (for example, 100 MHz is the bus speed of the Pentium® II Processor) and the graphics bus (the AGP bus operates at 133 MHz). Other buses can be expected to reach these speeds in the near future.

In order to achieve these speeds some buses are using a new architecture in which interconnect delays must be matched to each other. This "matching" means that printed circuit board designers must consider how manufacturing tolerances impact the mismatch between interconnect traces. This greatly increases the number of simulations required, and therefore the complexity of the design effort.

This paper shows there is a need to develop simulators capable of managing thousands of simulations, and that these tools must be able to present the results of these simulations in a format easily-understandable by designers.

II. INTERCONNECT DESIGN TRENDS

As system buses advance to speeds of 100 MHz and beyond, we are seeing a shift in timing architecture from a so-called "common-clock" timing mode to a so-called "source-synchronous" timing mode. This has a strong impact on the tools and techniques used to design these systems. In order to understand this impact it is first necessary to understand the differences between common-clock and source-synchronous modes.

A. Common-Clock

Fig. 1 shows an illustration of the common-clock timing mode. In this operating mode, a universal or "common" clock is generated elsewhere in the system and is used to launch data out of the driver and latch it into the receiver. The maximum operating speed for this type of bus is therefore limited by the sum of the output, interconnect and input delays as well as any routing skew between the two destinations for the clock. In other words, the minimum period (maximum frequency) of operation for this system is given by

$$\text{Period} = T_{\text{Driver}} + T_{\text{Interconnect}} + T_{\text{Receiver}} + T_{\text{Skew}} \quad (1)$$

where T_{Driver} is the driver's output valid delay (typically 5-10 ns), $T_{\text{Interconnect}}$ is the interconnect delay (typically 2-4 ns, depending on loading and fanout), T_{Receiver} is the receiver's input setup timing (typically 0-2 ns) and T_{Skew} is the skew between the clock at the driver and receiver (typically 0.5-1.0 ns). Using these ranges, we can estimate that common-clock mode will work well for switching speeds up to approximately 100 MHz, but another scheme will be needed as speeds exceed 100 MHz.

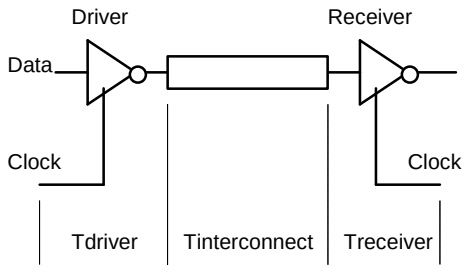

Fig. 1. Interconnect delays between a driver and a receiver, illustrating the common-clock timing mode.

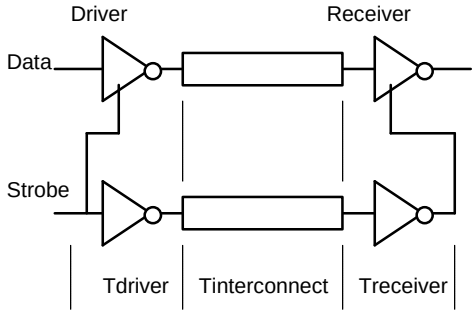

Fig. 2. Interconnect delay picture, including both data and strobe to illustrate the source-synchronous timing mode.

B. Source-Synchronous

As bus speeds increase, it becomes necessary to find a way to exceed the limit identified in equation (1). This can be done by generating the clock (more correctly called a "strobe" in this mode) locally and sending it as a separate signal along with the data. This is known as source-synchronous timing, and is illustrated in Fig. 2.

Source-synchronous timing allows the data and strobe delays to essentially cancel, and the speed of the bus is now given by

$$\text{Period} = (T_{\text{Driver}} + T_{\text{Interconnect}} + T_{\text{Receiver}} + T_{\text{Skew}})_{\text{Data}} - (T_{\text{Driver}} + T_{\text{Interconnect}} + T_{\text{Receiver}} + T_{\text{Skew}})_{\text{Strobe}} \quad (2)$$

Notice, however, that equation (2) indicates that the bus speed could theoretically approach infinity. This observation must, of course be false, but can we explain why? The key to explaining this lies in identifying all possible sources for mismatch between the data and strobe delays.

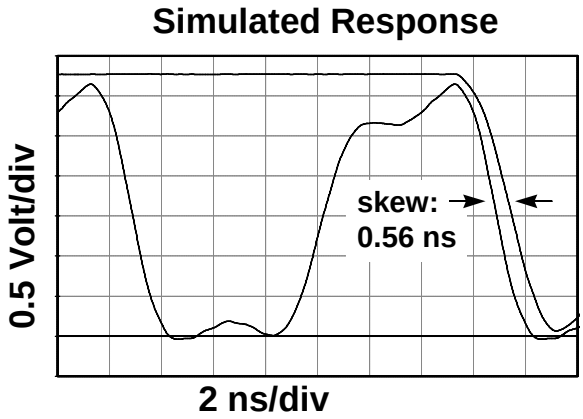

Fig. 3. Example of source-synchronous skew, showing the impact pulse width can have on the mismatch between two identical traces (strobe pulse = 7 ns; data pulse = 50 ns).

A better form, therefore, for equation (2) is

$$\text{Period} = (T_{\text{Data}} \pm \delta) - (T_{\text{Strobe}} \pm \delta) \quad (3)$$

where δ represents all of the uncertainty terms in these delay paths. These uncertainty factors include manufacturing tolerances in the printed circuit board, tolerances in the integrated circuit components, and additional effects caused by crosstalk and other noise sources. Some of the effects which must be considered are shown in TABLE I.

Note that in the best case this is limited by $\pm 2\delta$, so the designer must assume the worst-case operating speed, that is

$$\text{Period} = 2\delta \quad (4)$$

As a design problem, equation (4) is particularly interesting because δ represents terms which were previously called "second-order" effects. In earlier designs these terms were small enough that they could be ignored, but now they reflect nearly all of the design challenge.

For example, Fig. 3 shows what can happen if the strobe and data operate at different switching rates (common, because the data may easily contain several 1's or several 0's in sequence). Since the strobe's voltage level has not stabilized at the beginning of each new cycle, its interconnect delay is 0.5 nsec less than the data's interconnect delay. This additional 0.5 nsec of skew can be critical to 100 MHz source-synchronous designs.

At a personal level, this means the designer must become familiar with effects that were previously ignored.

TABLE I
SOME OF THE DESIGN PARAMETERS WHICH MUST BE CONSIDERED FOR COMMON-CLOCK AND SOURCE-SYNCHRONOUS DESIGNS

Common Clock	Source-Synchronous
• Driver strength	• Driver matching
• Receiver capacitance	• Receiver load matching
• Trace length	• Trace length matching
• Trace impedance and propagation velocity	• Trace impedance and velocity matching
	• Driver pullup and pulldown matching
	• Trace matching between even and odd (crosstalk) modes
	• Impact of pulse-width differences.

III. Interconnect Design Methodology

A. Design Complexity

To understand this complexity from the designer's point of view, we must consider the number of simulations necessary to guarantee sufficient performance.

To begin, consider the topology shown in Fig. 4. This is a cache design consisting of a processor, controller and 18 SRAM memory components (based on earlier 50 MHz designs using the Pentium® processor). The topology shows how a heavily-loaded common clock bus might be routed, and is a good example to show the impact of line length.

Fig. 5 shows the response of this system with the interconnect traces routed symmetrically. All components receive a well-shaped square wave with approximately 3 nsec delay, which is quite acceptable at 50 MHz.

Fig. 6 shows the response of the same topology when routed asymmetrically. In this case, the topology of Fig. 4 was modified so that the controller connects directly to the bottom row of SRAM's using a 1 inch trace. This is the type of routing that might occur if the router (manual or automatic) is trying to minimize interconnect lengths without knowledge of the resulting signal integrity. Fig. 6 shows that this asymmetry can nearly double the interconnect delay and can seriously degrade signal quality.

These figures show that even for a common-clock design the designer must consider several options and simulate those options before routing begins. Even at this level of complexity it may be necessary to simulate hundreds of cases to gain the understanding necessary to produce a working design. These cases must include analysis of the interconnect's performance over different line lengths (using the worst-case lengths expected in the final, routed design), different buffer impedances and rise/fall times (using the worst-case values expected due to the driver component's normal manufacturing tolerances) and different loading capacitances (using the worst-case expected due the the receiver component's normal manufacturing tolerances).

When a source-synchronous bus is being designed, the goal is to minimize the *difference* between the delays of two interconnect paths. For each case considered, the designer must compare two simulations in which the input variables were allowed to vary slightly (within normal tolerances). For example, when evaluating the impact of buffer strength on skew, the designer should simulate data and strobe using slightly different strengths, and then evaluate skew as a function of the *difference* between the two drive strengths.

For a source-synchronous design, the number of required simulations can be in the thousands.

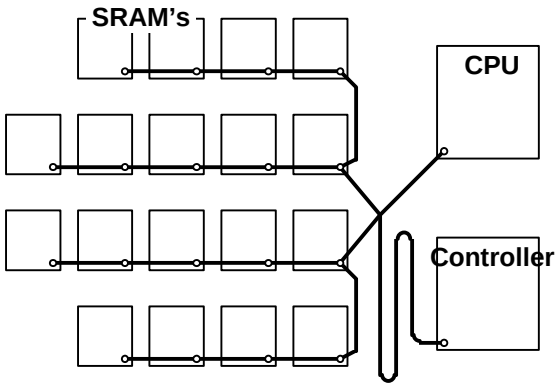

Fig. 4. Example of a heavily-loaded common-clock interconnect topology (example cache design from earlier 50 MHz systems).

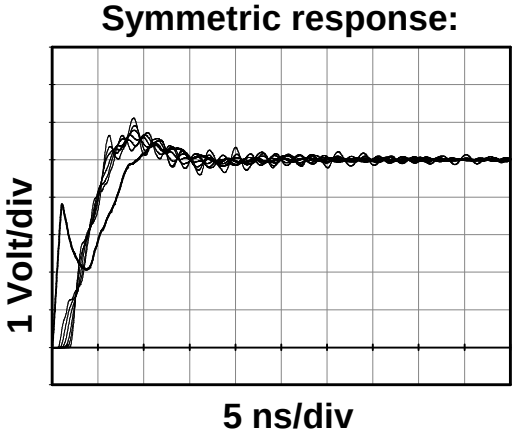

Fig. 5. Response of the topology in Fig. 4 when routed symmetrically. (The waveforms from all 20 components are overlaid in this plot; the waveform which has a step near Vcc/2 is the CPU, which is the driver in this example.)

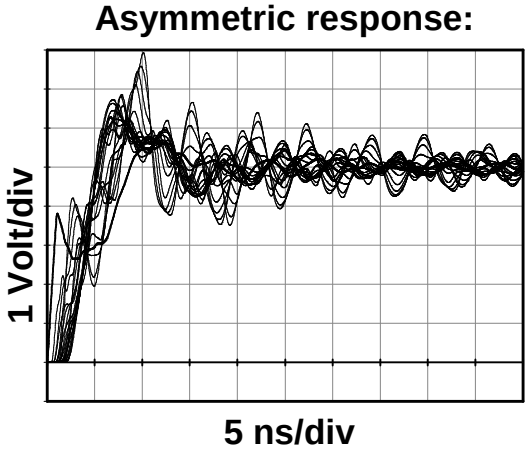

Fig. 6. Response of the topology in Fig. 4 when routed asymmetrically. (Driven by CPU).

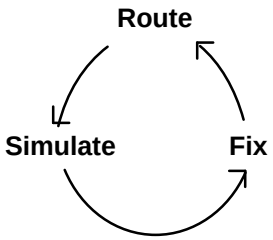

Fig. 7. Iterative design methodology (frequently non-convergent)

B. Dealing with thousands of simulations

When faced with the prospect of running thousands of simulations, most sensible designers will look for an easier way. They will most-likely revert to the design methodology shown in Fig. 7. In other words, the designer will simply route the board, and hope the simulator can detect and correct any problems. As Fig. 7 shows, this approach can be non-convergent. (The reason for this non-convergence is that it is usually impossible to fix "bad" traces without impacting "good" ones. Thus, after fixing several bad traces, the next round of simulations is likely to identify new "bad" traces).

A more desirable methodology is shown in Fig. 8. If implemented correctly, this methodology allows a design to be completed in a single pass, by relying on simulations that are run before the board traces are routed. The pre-route simulations are used to define routing "rules", which are then used to determine how the printed circuit board is routed, helping to ensure that all of the interconnects meet their performance reqirements on the first attempt.

However, this methodology is much more difficult to implement. It relies on a process called "sensitivity analysis", which can require more simulations than the designer can complete. To be effective, therefore, sensitivity analysis must be implemented as an automated feature in future simulation tools.

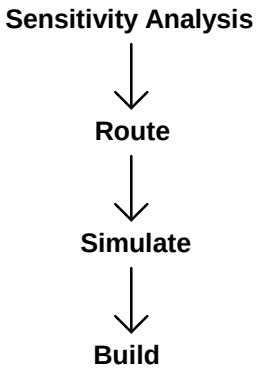

Fig. 8. "Single-shot" (ideally) design methodology.

IV. Sensitivity Analysis as a Design Tool

At this point we can see that the key to designing interconnects to operate at speeds in excess of 100 MHz lies in the ability to generate and analyze large numbers of simulations.

This section presents three examples of sensitivity analysis, showing three possible formats which may be used. These three formats pose two requirements on simulation tools:

- The tool must be able to run large numbers of simulations in batch mode, allowing design variables to be varied automatically.
- The tool must be able to present large amounts of simulation data in a format a *human* designer can understand. This format should be very visual.

The following plots are compiled from several past design projects. Plotting the data in these formats is not usually supported directly, and therefore requires the use of custom programs, usually written by the designers.

A. 3-Dimensional Sensitivity Analysis Plots

One type of sensitivity analysis is shown in Fig. 9. This type of analysis makes use of a three-dimensional plot, plotting performance (10% settling time, i.e. the time required for any oscillations to be damped to less than 10% of the signal amplitude) as a function of two design variables (driver strength and line length).

In actual use, it is not important for the designer to understand (or even to know) the definition of the term being shown on the vertical axis. It is only necessary to realize that "big numbers are bad; small numbers are good". From this analysis the designer can easily see that the board should be routed using a length of 4-5 inches for this trace.

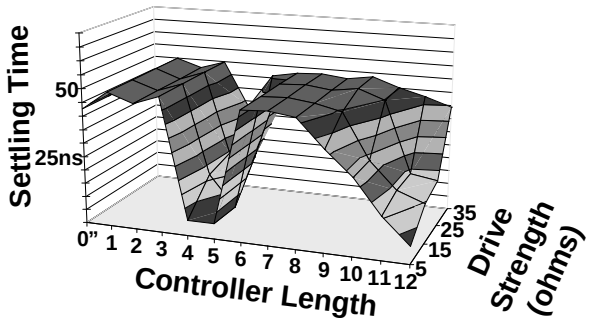

Fig. 9. Example of a 3-dimensional sensitivity analysis plot, showing the 10% settling time as a function of line length and driver strength, for the topology of Fig. 4.

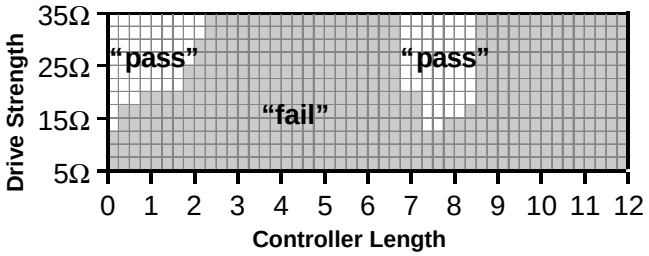

Fig. 10. Example of a solution-space plot.

B. Solution-Space Plots

Another type of sensitivity analysis, called a solution-space plot is shown in Fig. 10.

This type of analysis goes beyond the 3-dimensional plot by acknowledging that there may be several ways of specifying performance, and that all of these metrics impose requirements that must be met. A solution-space plot, therefore, is a way of testing whether the bus meets all of the required performance metrics, and plotting the results as a function of any two design variables. The example in Fig. 10 shows the pass/fail test results of the interconnect in question plotted as a function of driver strength and line length.

In this case the designer can easily see that the board should be routed using a trace length of 7.5-8.5 inches, and that components should be chosen which have output impedances greater than 20Ω.

C. Monte-Carlo Solution-Space Plots

The third type of sensitivity analysis, shown in Fig. 11 is a monte-carlo solution space plot. This type of plot is similar to the solution-space plot, except that all of the design variables have been allowed to vary randomly. After the simulations have been completed, their results are re-sorted and plotted against any two of the design variables. Points at which the interconnect meets all of its performance requirements are indicated with a white symbol; points at which the interconnect fails to meet any of its requirements are indicated with a black symbol.

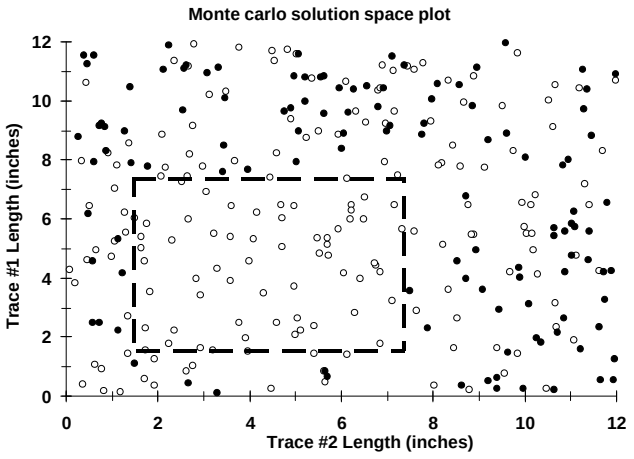

Fig. 11. Example of a monte-carlo solution-space plot. Light symbols indicate "pass"; dark symbols indicate "fail".

In use, a designer can view this plot and understand that the two traces should both be routed within the range of 1.5-7.5 inches. This technique is similar to the solution-space plot shown in Fig. 10, but has the added benefit of allowing all input variables to be varied randomly, helping to ensure better coverage of the design space.

All of these techniques help the designer understand simulation results without reviewing gross amounts of raw simulation data.

V. Conclusions

In the future, as bus speeds continue to increase beyond 100 MHz, designers will find it necessary to generate thousands of simulations for a single design.

Future simulation tools must therefore focus on the ability to handle these large numbers of simulations. Specifically, these tools must be able to run large numbers of simulations in batch mode, allowing parameters to be varied automatically without requiring human intervention, and post-processing the results into an easily-readable format.

In short, the goal is to create pictures which help the designer visualize and understand performance trends.

VI. References

[1] H. W. Johnson and M. Graham, *High Speed Digital Design: A Handbook of Black Magic*, Englewood Cliffs, NJ: Prentice-Hall, 1993.

[2] Intel Corporation, *Accelerated Graphics Port (AGP) Platform Design Guide*, published on the worldwide web at http:\\www.agpforum.org, June 1997.