Treat pc-board traces as transmission lines to specify drive buffers

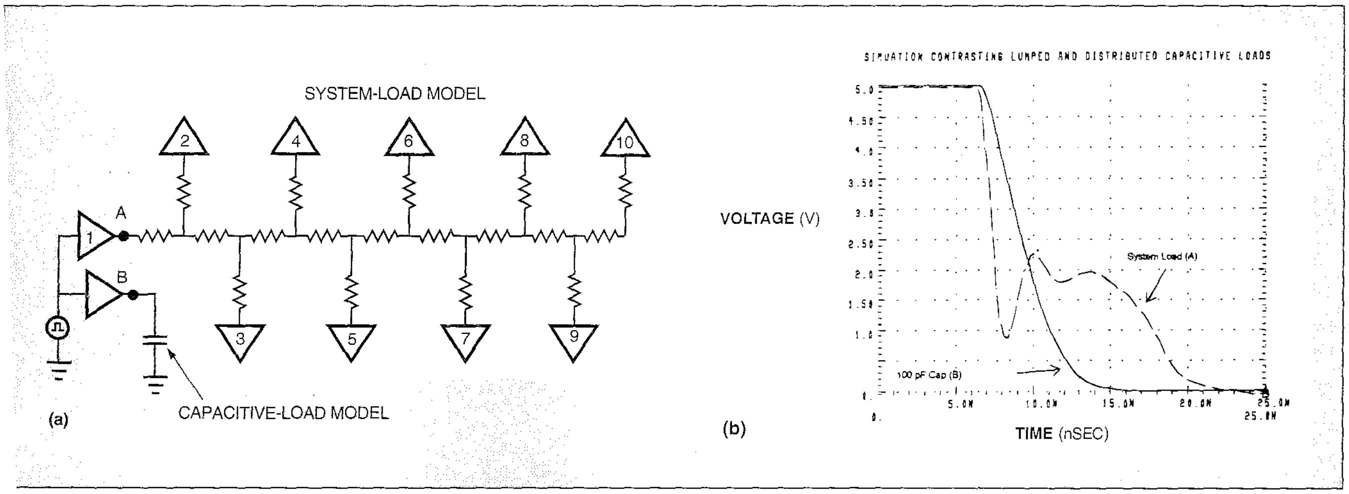
Donald Telian, Intel Corp

The Peripheral Component Interconnect (PCI) specification simplifies the design of buffer circuitry in high-speed interface systems. By using transmission-line concepts and employing ac specifications in place of dc values, PCI lets designers more accurately define a buffer's drive capability.

By employing transmission-line models and terms, the PCI specification developed by Intel Corp can thoroughly characterize the dynamic effects that result when you switch large distributed loads with a underspecified drive buffer. Because many designers oversimplify this problem, designing buffers with appropriate drive strength turns into a difficult task. Historically, designers have assumed that buffers work into simple capacitive loads. This assumption created few problems when system clock rates loafed along in the 1-MHz range. In today's world of 30-MHz-plus clock rates, however, real system loads are far from purely capacitive. In fact, even pc-board trace lengths can dramatically change the makeup of a buffer's load.

Viewing the buffer load as a simple capacitor can lead to many erroneous conclusions. The most common mistake designers make is to assume that an interconnect with many loads is like a large capacitor, which must therefore require a buffer with a large drive capability. It turns out that this is not the case, and designers must understand the transmission-line mechanisms involved here. To better understand the difference in driver response between a capacitive load and a true system load, consider **Fig 1a**.

Here, a system load is actually made up of numerous devices interconnected through traces that you can model as transmission lines. Aside from their clamping characteristics, the receiving devices are predominately capacitive. However, the system model represents a distributed capacitance and yields waveforms



g 1—A system load is made up of multiple devices interconnected through pc-board traces, which you can treat as transmission lines (a). iis distributed capacitance yields waveforms very different from those of the lumped-capacitance model (b).

very different from those of the lumped-capacitance model **(Fig 1b)**.

Fig 1b is the output of an HSpice simulation of the models in **Fig 1a**. To contrast the difference between the two models, identical drivers applied identical signals to points A and B in **Fig 1a**. The generator synchronized the two drivers by applying the same square wave to their inputs. For this simulation, the total capacitance at each of the 10 devices is 10 pF. Neglecting trace capacitances, the total capacitance is 100 pF. However, notice the marked difference between the waveform of the 100-pF distributed-capacitance system model and that of the 100-pF lumped-capacitance model.

Neglecting actual system electrical effects can cause serious problems in the design of a pc-board local bus or any high-speed digital interconnect. A closer look at **Fig 1b** shows that the actual waveform passes the 1.5V mark (the switching threshold) 6 nsec after the time predicted by the lumped-capacitance model. For a 33-MHz cycle time, a 6-nsec oversight could cause a signal to miss its entire setup time. Furthermore, the capacitive-load model does not show any of the odd signal properties inherent in an actual system. For example, why does the system waveform in **Fig 1b** sit at 2V for 5 nsec? There's a simple way to answer this and related questions: Dispense with the capacitive-load model and adopt a transmission-line model of the high-speed system interface.

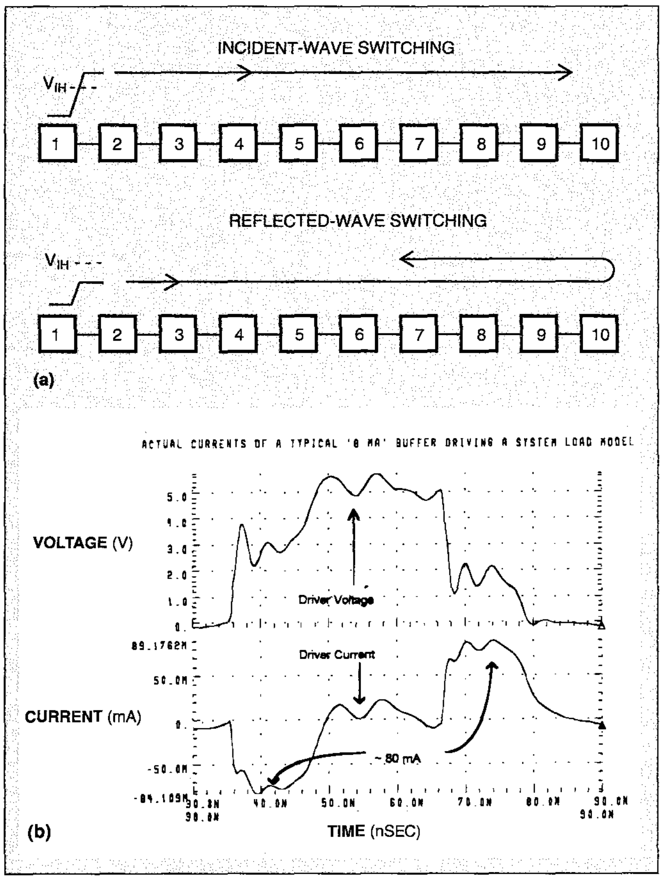


Fig 2—Because the drive-level requirement is higher in incident-wave switching schemes (a), the technique places a higher stress on system components. The output performance for an 8-mA ASIC buffer (b) shows a large disparity between actual ac values and dc specifications.

Modeling board traces

Transmission lines are typically modeled as an impedance and a time delay. It is important for digital designers to develop at least an intuitive understanding of these two properties and how they might affect a design. Also, designers should keep in mind that the environment of interest is pc-board traces and integrated circuits, not coaxial cables or power lines.

A printed-circuit trace's impedance begins as a function of the fabrication of the board itself. Today's fabrication methods commonly yield trace impedances anywhere from 50 to 110Ω. The primary factors causing this variation are the width of the trace and the distance from the trace to a ground plane. If a trace is wide and close to a ground plane, it is more capacitive and has an impedance close to 50Ω. Conversely, if the trace is narrow and a good distance from a ground plane, it is more inductive and its impedance approaches 110Ω. Now consider what happens to trace impedance when the trace is driving multiple ICs. Because the extra load of each device is primarily capacitive, the trace impedance decreases.

A transmission line's time delay is easier to understand. It takes time for the charge an IC supplies to travel down a pc trace. For example, if driver 1 in **Fig 1a** switches from low to high, time will pass before this transition reaches load 10. You must take this time, or system-propagation, delay into account for boards having many loads or long traces.

Follow some switching techniques

Given that a signal must propagate down the board trace, two questions arise: Is the signal strong enough to change logic levels at all the devices as it passes by? And what will happen when the signal reaches the end of the trace? Let's address the first question first.

Fig 2a shows a simplified representation of two 10-load systems. In both systems, device 1 drives the interconnect from low to high. In the top system, the driver is strong enough at incidence (when it first starts to drive) to pull the voltage above the high-level input voltage (V_{IH}) of all the other components. Component 2 is the first to see this valid voltage level, followed by component 3, and then 4, and so on. In this type of implementation, component 10 is the last device to see a valid high level after one full system-propagation delay, which is the time required for the signal to travel from one end of the trace to the other. Because the incident voltage wave was strong enough to switch each device as it passed by, this switching technique is called incident-wave switching.

At first glance, this implementation seems ideal because it accomplishes the switching task in the shortest amount of time. But consider the implications. The extra devices on the pc-board trace add capacitance and lower the impedance presented to the driver. In such configurations, it's not uncommon for the loaded impedance to drop to 30Ω. Now, when you consider the load as a transmission line, you can figure out exactly how the system will behave under these circumstances. For example, the moment a 5V device switches from low to high, it creates a voltage divider between its own output impedance and the transmission-line impedance. If the line is 30Ω, a 20Ω driver would drop 2V inside the component and inject a 3V incident wave into the system. The current surge the driver sources during this process would be 100 mA.

The problem with this 100-mA surge current becomes apparent when you try to use it as a source for a complex device. A typical 32-bit interface might have 40 signals trying to switch instantaneously. Thus, up to 4A of dynamic current passes through the device, possibly in as short a time as one nanosecond. This type of current surge is extremely difficult to decouple. The surge causes spikes on internal bond wires, increases EMI, causes crosstalk in and out of the package, and is the reason why most strong buffers come in 8-driver packages. Furthermore, the 20Ω driver requires a lot of silicon space and will get fairly hot at high frequencies. Note also that in a TTL environment, these problems are generally twice as great on high-to-low transitions because the incident-wave magnitude must be even larger. All things considered, this switching method is not preferable from a device standpoint.

In addition to being hard on components, incident-wave switching also has negative effects in the system. These effects are related to the second question: What happens to the propagating signal when it reaches the end of the trace? The end of a normal, unterminated trace presents a high impedance and leaves the signal no options; it must turn around and go back. The transmission-line world calls this signal return a reflection. At the site of the reflection, the wave begins propagating back to the driver, doubling the voltage on the line. If the incident-wave voltage is large (as in the case of

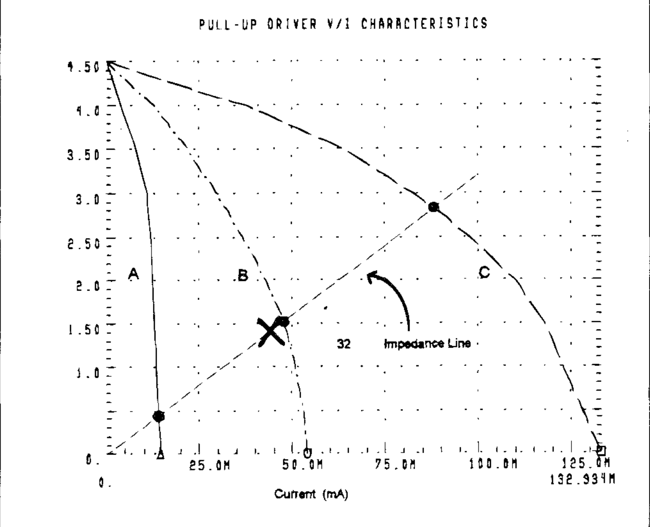


Fig 3—You can quickly estimate a driver's ability to meet the specified V_{OH} (ac) and I_{OH} (ac) levels by examining its V/I curve.

incident-wave switching), this reflection quickly becomes unwieldy. Designers often place terminating devices at the ends of traces. However, this technique requires more components, and the terminating devices draw too much dc power and are difficult to tune.

Introducing reflected-wave switching

The PCI specification takes advantage of these inherent reflections by introducing reflected-wave switching. Reflected-wave switching achieves full voltage transitions from currents that are much lower than those required in incident-wave switching schemes.

To understand how reflected-wave switching works, consider the lower example in **Fig 2a**. The driver (device 1) needs to switch on incidence only half the voltage required for incident-wave switching. This lower voltage then propagates past device 2, then past 3, and so on, until it reaches device 10. At device 10, the reflection doubles the voltage level, causing device 10 to be the first to experience a valid high level. The wave then propagates back past device 9, then past 8, and so on, until it is absorbed in the low impedance of the driver. For reflected-wave switching, the last receiver to see a valid voltage level is device 2. To ensure proper operation, it is essential that you budget time for the reflected-wave propagation.

Besides taming the reflection problem, reflected-wave switching also cuts driver size and surge currents in half. In fact, reflected-wave methods can switch an entire 10-load, 33-MHz PCI subsystem with what the high-integration ASIC world might call a 12-mA (dc) pull-down current and a 6-mA (dc) pull-up current. However, because you're trying to guarantee an ac phenomenon, PCI departs from standard dc specification methods.

Choosing an appropriate driver

Reflective-wave switching lets you connect highly integrated devices to a local bus without using extensive glue logic, but you still have to choose a buffer. Standard ASICs offer a variety of choices with values ranging from 1 to 24 mA and various slew options and pull-up voltages. But how do these choices map to a dynamic switching environment? Which one will guarantee reliable reflected-wave switching, or should you just use the buffer with the largest available current value? There are straight-forward answers to these questions, but once again, you'll have to adjust the way you look at the design problem.

First, understand that buffers are normally specified by their dc capacity. For example, a 24-mA buffer can sink 24 mA of direct current without raising its output voltage above V_{OL} (the guaranteed low-state output voltage). Historically, this specification was important because inputs sourced appreciable current and using terminations and pull-up resistors was common. However, today's CMOS inputs have leakage currents on the order of only a couple of microamps or even nanoamps. Consequently, the dc specification has little value today and has become an inappropriate quantifier of ac drive strength.

To illustrate the disparity between actual ac values and dc specifications, consider **Fig 2b**, which shows waveforms of a typical 8-mA ASIC buffer driving the 10-load system model of **Fig 1a**. The top waveform is the voltage at the driver's output; the bottom waveform is the output's dynamic current. This waveform

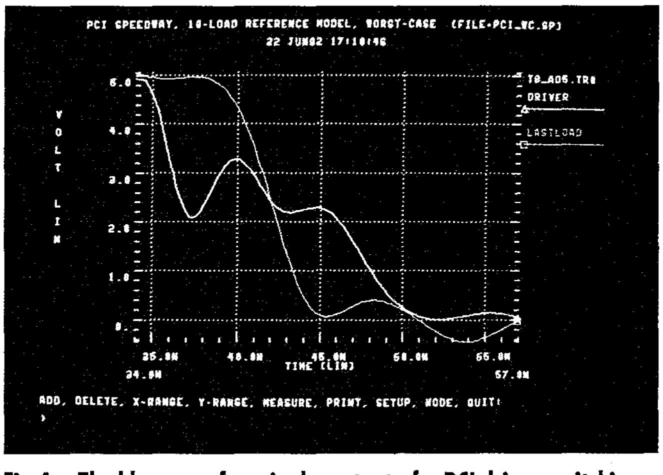


Fig 4—The blue waveform is the output of a PCI driver switching a 10-load system model like that of Fig 1a. The signal propagates through the system; the waveform observed at device 10 is shown in red.

plainly shows that the actual ac switching currents are approximately 80 mA—a full order of magnitude greater than what you might expect from an 8-mA buffer.

Note also that the signals in **Fig 2b** spend about one third of the cycle, or 10 nsec, at the higher dynamic-current level. Indeed, at a 30-nsec cycle time, the signals spend little time in a static dc condition. Thus, the bulk of the cycle time is exhausted in an unspecified condition and hardly understood. When typical operating speeds were only 1 MHz, you could overlook that 10-nsec dynamic operation because it represented only 1% of the cycle. The other 99% of the cycle was carefully specified by dc parameters. But at today's higher speeds, these figures no longer hold, and correct buffer design (as well as reliable operation) is largely a function of understanding and specifying the ac characteristics of an interconnect, which is difficult to do with only dc specifications.

Specifying in reliability

To overcome this problem, the PCI specification lets you establish the minimum dynamic-current levels required to ensure reliable ac operation. New quantities I_{OL} (ac) and I_{OH} (ac) guarantee adequate switching current for high-to-low and low-to-high transitions, respectively. When paired with the required transmission-line step voltages, V_{OL} (ac) and V_{OH} (ac), these values guarantee successful reflected-wave switching. You can derive I_{OL} (ac) and I_{OH} (ac) for any interconnect based on the interconnect's input voltages and the system impedance.

Simulation shows that 0.8V is a sufficient noise margin for transient voltage ringing. This ringing occurs at the point of reflection (device 10 in **Fig 2a**) and must be within the allowed margin by the time the wave traverses all the loads on the return trip. Because V_{IH} is always 2V, a 100% reflection must achieve a level of 2.8V. This implies that the driver must supply an initial step of only 1.4V. Consequently, V_{OH} (ac) is 1.4V.

To determine the required value for I_{OH} (ac), use I=V/R, where V=1.4V, and R is the acceptable loaded impedance of the pc-board trace. PCI designers found 32Ω to be an acceptable lower bound for loaded transmission-line impedance for a variety of layouts. Equations in the PCI specification enable you to calculate the loaded trace impedance for a given layout. In the past, interconnects with multiple loads used a lower bound of 25Ω when the average unloaded pc-board trace impedance was 50Ω. However, today's average unloaded trace impedance is closer to 65Ω because of the industry-wide use of narrower traces.

Given values for R and V_{OH} (ac), you can determine the required dynamic current level as I_{OH} (ac)=V_{OH} (ac)/R, or 1.4V/32Ω=44 mA. Consequently, for a driver to comply with PCI, it must be able to deliver at least 44 mA when its load drops to 32Ω. This requirement ensures that the initial voltage step is at least 1.4V, which will reflect to 2.8V and provide a 0.8V ac noise margin.

You can quickly determine a driver's ability to meet the calculated V_{OH} (ac) and I_{OH} (ac) levels by examining its V/I characteristic—normally a curve that describes the output's current capabilities across all voltages of interest. Such curves have been used for years to describe the characteristics of transistors and diodes and serve well to quantify the analog effects of digital signals. **Fig 3** shows the V/I characteristics of three typical 5V CMOS pull-up drivers, labeled A, B, and C. A large "X" marks the PCI-specified V_{OH} (ac) and I_{OH} (ac) values for the three drivers and establishes a baseline for the exact strength of all PCI drivers.

In **Fig 3**, driver B is the optimal PCI driver because it meets the required specification without overdriving the interconnect. The plot shows that driver B will slightly exceed the specification by supplying about 50 mA at 1.4V. To illustrate the inadequacies of drivers A and C, a 32Ω impedance line is superimposed on the CMOS curves of **Fig 3**. Graphical techniques show that each driver will switch (at incidence) the voltage predicted by the intersection of its V/I curve and the 32Ω-load line. Driver A will switch to only 0.5V at incidence, and its reflection could achieve only 1V. A driver this weak would require multiple system-propagation delays to achieve a stable logic high and would not work at the required frequencies.

The intersection of driver C's V/I curve shows that it would switch almost 3V at incidence. This driver certainly satisfies the 44-mA requirement but is not optimized for a reflected-wave interconnect. It would require too many grounds and too much silicon. This type of driver could probably never be integrated into a low-cost, highly integrated device.

Because driver B sources enough current to satisfy system needs without overdriving the system, it is the optimal driver. Designers could easily integrate this driver and could connect peripheral chips to a local bus without needing glue logic. Reflected-wave switching, guaranteed by dynamic-current levels, provides integrated devices a low-current mechanism to drive an entire high-speed I/O subsystem.

Simulation proves the concepts

These concepts could not have been reduced to specifications or implemented in hardware without first thoroughly testing them through simulation. The advanced features of HSpice from Meta-Software

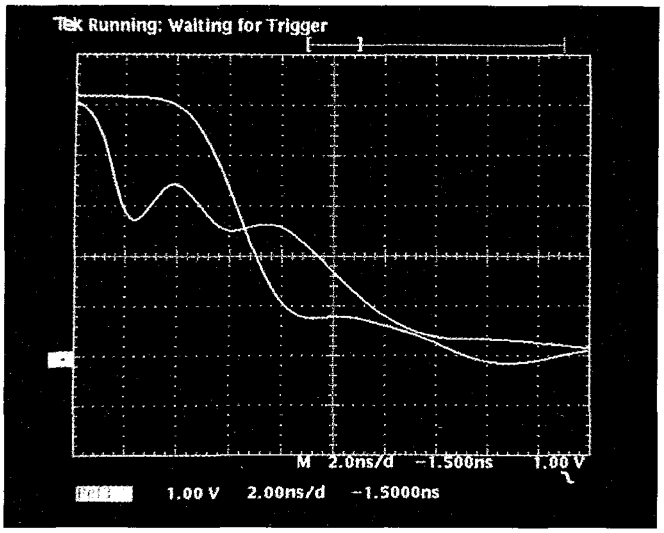


Fig 5—Signal measurements on a board made up of the exact same buffer, loads, and trace dimensions as those of the circuit simulated in Fig 4 show the high accuracy of the pc-board simulation.

(Campbell, CA) were particularly well suited for this task. Intel engineers logged more than 5000 hours of HSpice simulation in the process of defining the PCI specification and ensuring its robustness. **Fig 4** shows a plot of one PCI simulation.

The blue waveform is the output of a PCI driver switching a 10-load system model similar to the model of **Fig 1a**. Because reflected-wave switching is in use here, the output switches the trace only half way to the desired potential of 0V. The signal propagates through the system, and the waveform observed at the end of the trace (device 10 in **Fig 1a**) is shown in red. Because the trace is intentionally unterminated, the signal reflects and achieves the desired 5V transition. After reflecting, the signal propagates back to the driver (device 1) causing it to be the last device to observe the full transition.

Careful simulation shows that this round-trip propagation requires 10 nsec, worst case. The PCI specification refers to this time as T_{PROP} and builds it into the base cycle-time parameters. This time delay clearly arises because of the transmission-line nature of the long trace. HSpice lets you simulate these effects by providing a proprietary element that derives a lossy transmission-line model from the fabrication geometry of the pc-board trace. Obviously, ignoring this 10-nsec phenomenon would jeopardize the operation of the system, because 10 nsec equals a third of the cycle time.

To prove the validity of the electrical design and simulations, PCI designers created an interconnect topology they called the PCI Speedway—a recommended layout approach with built-in signal integrity. Further, the designers created an actual pc

SPECIFYING BUFFERS FOR HIGH-SPEED DIGITAL SYSTEMS

board—an implementation of the PCI Speedway in hardware. The designers fully tested the board's complete 10-load, 33-MHz PCI electrical design prior to the specification's publication. More than 100 types of ASICs and discrete buffers were electrically tested on the PCI board.

Measuring the actual hardware proved the accuracy of the simulations. **Fig 5** shows a signal measured on the Speedway board. This signal provides a hardware test of the exact same buffer, loads, and trace dimensions of the circuit simulated in **Fig 4**. Careful comparison of **Figs 4** and **5** reveals the accuracy of the simulation.

Demonstrated correlation, as **Fig 5** showed, proved that the concepts set forth in the PCI specification are both practical to implement and reliable long before actual PCI systems were built. In addition, the notion of predicting ac drive performance from a buffer's V/I characteristics was thoroughly tested and verified. This process was accomplished by first obtaining the physical device's V/I curves on a standard curve tracer, then attaching the device to the Speedway hardware, and finally verifying that the PCI driver gave the correct step voltage into the known impedance. Being able to predict the minimum step realized by a PCI driver was critical to ensure reliable reflected-wave switching. EDN

Author's biography

Donald Telian is interconnect-design-team manager in the Integrated Microcomputer division of Intel Corp in Folsom, CA. In this position, he coordinates divisional interconnect-simulation efforts. Don holds a BSEE from the University of California at Santa Barbara and has worked for various electronic companies since 1986. He has been integral in defining electrical interconnect standards for both Hewlett-Packard's Precision Architecture and the component specification of PCI.