VLSI Design, Vol. 7, No. 1 (1998) 111-129

High Performance, Point-to-Point, Transmission Line Signaling

André DeHon Thomas F. Knight, Jr.
andre@ai.mit.edu tk@ai.mit.edu
MIT Artificial Intelligence Laboratory
NE43-791, 545 Technology Sq., Cambridge, MA 02139
Phone: (617) 253-5868 FAX: (617) 253-5060

June 30, 1995

Abstract

Inter-chip signaling latency and bandwidth can be key factors limiting the performance of large VLSI systems. We present a high performance, transmission line signaling scheme for point-to-point communications between VLSI components. In particular, we detail circuitry which allows a pad driver to sense the voltage level on the attached pad during signaling and adjust the drive impedance to match the external transmission line impedance. This allows clean, reflection-free signaling despite the wide range of variations common in IC device processing and interconnect fabrication. Further, we show how similar techniques can be used to adjust the arrival time of signals to allow high signaling bandwidth despite variations in interconnect delays.

This scheme employed for high performance signaling is a specific embodiment of a more general technique. Conventional electronic systems must accommodate a range of system characteristics (*e.g.* delay, voltage, impedance). As a result, circuit designers traditionally build large operating margins into their circuits to guarantee proper operation across all possible ranges of these characteristics. These margins are generally added at the expense of performance. The alternative scheme exemplified here is to sample these system characteristics in the device's final operating environment and use this feedback to tune system operation around the observed characteristics. This tuning operation reduces the range of characteristics the system must accommodate, allowing increased performance. We briefly contrast this sampled, system-level feedback with the more conventional, fine-grained feedback employed on ICs (*e.g.* PLLs).

Acknowledgments: This research is supported in part by the Advanced Research Projects Agency under contracts N00014-87-K-0825 and N00014-91-J-1698. This material is based upon work supported under a National Science Foundation Graduate Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

In this paper, we address the issue of high-performance, point-to-point transmission-line signaling. Our objective is to achieve low transmission latency and high signaling bandwidth with a design which is economical in real-estate and power consumption while remaining compatible with commodity IC technology.

For large VLSI systems, inter-chip signaling can account for a significant fraction of the operational cycle time. The delay between a pair of ICs can be decomposed into three components:

1. *output delay* – delay through the output pad to drive the large, external capacitance associated with any component pad
2. *signal propagation delay* – the time required for a signal to propagate across the interconnect media from the source to the destination
3. *input delay* – delay through the input pad while the signal is being sensed and level-restored for internal IC consumption

In this paper, we specifically address the issue of minimizing signal propagation delay across transmission line interconnect. Note that interconnect is best modeled as a transmission line any time when the propagation delay across the interconnect is comparable to or greater than the rise time or fall time of the signal. That is:

$$t_{pd} \geq t_r \quad (1)$$

The transmission line propagation delay (t_{pd}) is determined by the materials in use and the physical interconnect length (l):

$$v = \frac{1}{\sqrt{\mu\epsilon}} = \frac{c}{\sqrt{\mu_r \epsilon_r}} \tag{2}$$

$$t_{pd} = \frac{l}{v} \tag{3}$$

For most available interconnect technologies $\mu_r = 1$ and $2 \leq \epsilon_r \leq 5$. With a relative dielectric constant (ϵ_r) of four ($v_{prop} = \frac{c}{2} = 15$ cm/ns), which is common among PCB technologies, and fast edge rates ($t_r \leq 0.5$ ns), traces over a few centimeters begin to exhibit transmission line effects. For long interconnects (tens of centimeters), propagation delay becomes the dominant fraction of inter-chip communication delay.

The minimum transmission line propagation delay, however, is only achieved when the transmission line is properly terminated so that no reflections occur and the line settles to the desired voltage in one propagation time. Process variation in IC and PCB fabrication make the termination matching difficult to achieve. Termination integrated on-chip is desirable to avoid the area and cost associated with external termination devices but on-chip resistances vary widely with IC processing, operating voltage, and temperature. Further, the characteristics of the interconnect media itself can vary. To avoid these problems, we develop a technique which allows an output driver to examine the voltage on the line during signaling and servo the impedance to match the attached interconnect. This technique requires no external termination devices to achieve clean impedance matching and allows a single IC to match cleanly across a wide-range of interconnect impedances.

This paper also addresses the issue of increasing signaling bandwidth on our transmission line interconnect. In this domain, two key properties limit signaling bandwidth:

- *signal deformation* – limited rise and fall times along with dispersive effects in circuits and interconnect spread out data bits.
- *skew and delay variation* – uncertainty in the propagation delay through components or interconnect must be accommodated by spacing the data bits to accommodate the entire range of possible transition timings.

To achieve reliable, high bandwidth signaling over transmission line interconnect, our techniques sense the delays actual seen on a fabricated component in system. On-chip delay is then adjusted to make the total propagation delay between ICs conform to tighter timing constraints. Together, these techniques allow us to remove much of the uncertainty and variation associated with signal transmission and, consequently, significantly reduce the inter-bit separation necessary for reliable inter-chip signaling. As with the impedance matching, this technique also allows a system to tolerate a larger variance in interconnect characteristics while still achieving reliable, high bandwidth operation.

Most CMOS circuit designers are familiar with the practice of designing circuitry to compensate for the wide variations associated with silicon processing. These adjustable termination and timing techniques take the strategy one step further allowing the component to compensate for variations in its external environment. The key theme here is to measure system characteristics and then employ tuning circuitry to bring the characteristics into a tight and favorable operating range.

Bringing these techniques together, we present a design for adaptable i/o pads engineered for high-performance, point-to-point transmission-line signaling. Section 2 presents an overview of the signaling strategy. We show a low-voltage swing, matched-impedance output pad for driving series-terminated transmission lines along with a complimentary receiver in Section 3. We introduce a technique in Section 4 for capturing dynamic timing information in response to signaling events. Section 5 details how the series impedance is adjusted in-system so the source drive impedance matches the impedance of the attached transmission line. Section 6 reviews techniques for on-chip delay adjustment and Section 7 shows how the timing-extraction and delay adjustment techniques facilitate the retiming of inter-chip communications. Section 8 describes how these techniques impact the testing of high performance interconnect. In Section 9, we present highlights of a prototype i/o pad which incorporates many of the techniques described in this paper. In Section 10 we highlight limitations of the techniques presented here before concluding in Section 11.

2 Signaling Strategy

To meet the needs of point-to-point signaling with high speed and acceptable power, we utilize a series-terminated, low-voltage swing signaling scheme which uses on-chip termination and employs feedback to match termination and transmission line impedances. For the purposes of the following discussion, we focus on a CMOS integrated circuit technology.

Low-voltage swing signaling is motivated by the desire to drive the resistive transmission line load with acceptable power dissipation. We see in Equation 4 that power is quadratic in signaling voltage. In the designs which follow, we specifically consider signaling between zero and one-volt. Limiting the voltage swings to one-volt saves a factor of 25 in power over traditional five-volt signaling (*i.e.* with a 50Ω transmission line, $P_{drive} = 250\text{mW}$ with five-volt signal swings and $P_{drive} = 10\text{mW}$ with one-volt swings).

$$P_{drive} = \frac{(\Delta V_{line})^2}{R} \qquad (4)$$

Recall that a properly series terminated transmission line will present a load of $R = 2Z_0$ to the driver since the transmission line impedance (Z_0) occurs in series with the termination resistance.

To achieve one-volt signaling, we provide components with a one-volt power supply for the purpose of signaling. This frees the individual components from converting between the logic supply voltage and the signaling voltage level. Any power which must be consumed in the process of generating the one-volt supply is dissipated in the power supply, and not in the individual ICs.

Series termination offers several advantages over parallel termination for point-to-point signaling. First, we can integrate the termination impedance into the driver. In the parallel terminated case, we needed to drive the voltage on the transmission line close to the signaling supply rail. The effective resistance across the driver between the supply rails and the driven transmission line must be small compared to the transmission line impedance, Z_0, in order to drive the transmission line voltage close to the signaling supply (See Figure 1). In a CMOS implementation, this means that the size of the transistors implementing the final driver must have a large W/L ratio to make the resistance small. As a consequence, the final driver is large and, therefore, has considerable associated capacitance, C_{driver}. The larger capacitance, of course, increases the delay required to buffer internal logic signals to drive the final pad output stage. It also means that the charging power, P_{charge} (Equation 5), will be large. In contrast, the series terminated driver can use a higher-impedance driver. The higher impedance of the series terminated driver allows it to have a smaller W/L ratio and hence smaller C_{driver}, resulting in a lower output delay and requiring less power to drive the output.

$$P_{charge} = \frac{1}{2} C_{driver} (\Delta V_{driver})^2 f \qquad (5)$$

Additionally, the series terminated configuration gives us the opportunity to use voltage feedback to adjust the on-chip, series termination to match the transmission line impedance. We expect both the transmission line impedance and the conductance of the drive transistors to vary due to process variations. By monitoring the stable line voltage during the round-trip transit time between the initial transition at the source end of the transmission line and the arrival of the reflection, a controller can identify whether the driver termination is high, low, or matched to the transmission line impedance. With a properly terminated series transmission line, we expect the voltage to settle half-way between ground and the signaling supply during the first round-trip transit time. If the voltage settles much above the half way point, the drive impedance is too low. If the voltage settles much below the half-way point, the driver impedance is too high (See Figure 2). By monitoring the voltage at the pad, the system can adjust the drive impedance until it matches the line impedance. This allows the integrated circuit to compensate for process variation in both the silicon processing and PCB manufacture.

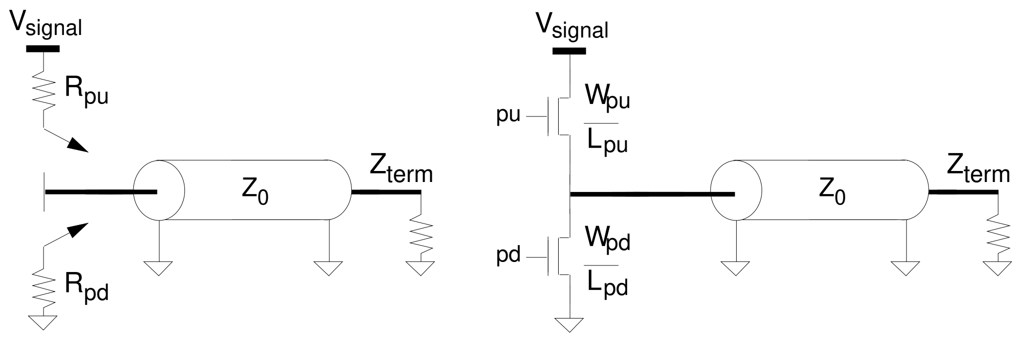
3 Signaling Circuitry

In this section we present driver and receiver circuitry which facilitate low voltage signaling and impedance adjustment.

3.1 Driver

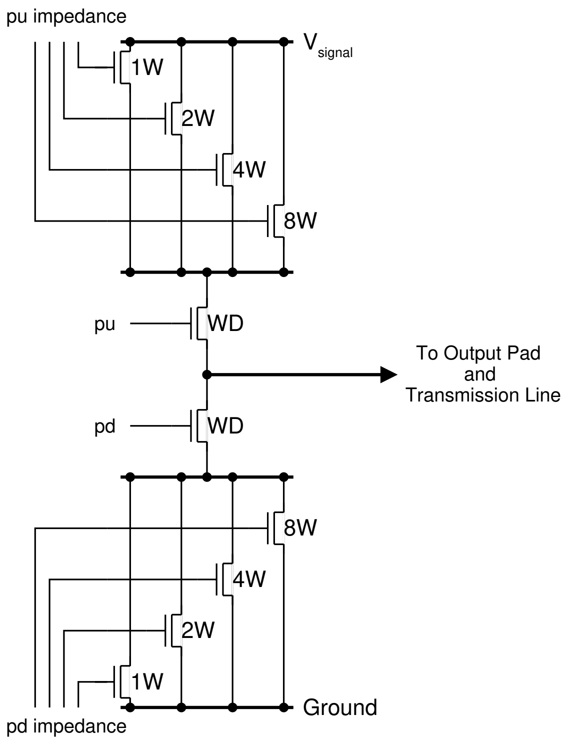
To control the output pad impedance, the output driver is connected to the high and low signaling supplies through adjustable impedance networks. As shown in Figure 3, a set of exponentially sized drive transistors form the adjustable impedance network. The impedance control drivers can be enabled via digital control lines from a scan-loaded control register and serve as a D-to-A network for the pad drive resistance.

Gabara and Knauer suggest a similar scheme which places only the set of exponentially sized transistors between the signaling supplies and the output pad [6]. An AND gate preceding the D-to-A network serves to combine the logical drive value with the impedance selection.



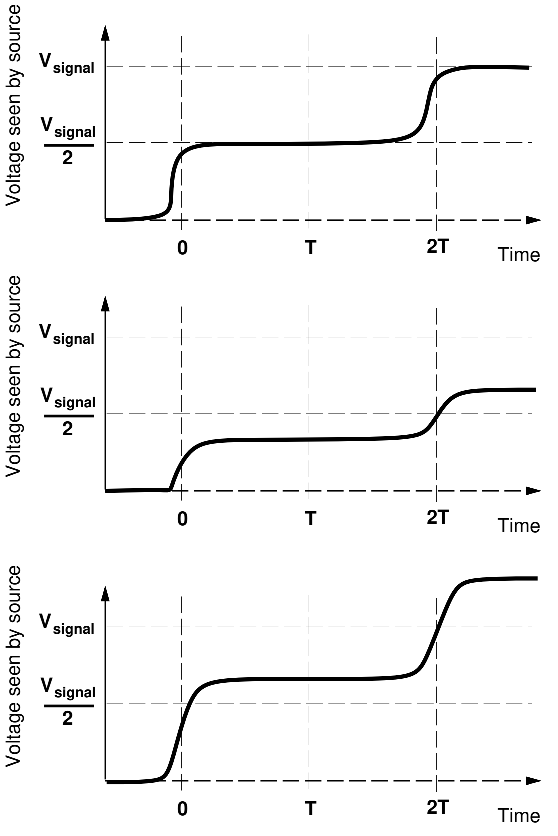
Shown here is the basic CMOS transmission line driver. Shown at right is the basic driver. Shown on the left is a simplified model of the driver making explicit the fact that each transistor, when enabled, can be modeled as a resistor of some resistance determined by the transistor's W/L ratio and process parameters.

Figure 1: CMOS Transmission Line Driver



Shown above is a digital controlled-impedance driver after [5]. The digital values on *pu_impedance* and *pd_impedance* enable the parallel impedance control transistors. Drive transistors are placed in series between the impedance control networks and the output pad. The desired signaling voltage is connected to the pad by enabling the appropriate drive transistor. The digital impedance controls remain static during normal operation. *N.B.* This design assume the signaling supply (V_{signal}) is lower than the high logic value on *pu* and the *pu_impedance* lines making it feasible to employ NMOS transistors for the pull-up impedance network.

Figure 3: CMOS Driver with Adjustable Drive Impedance

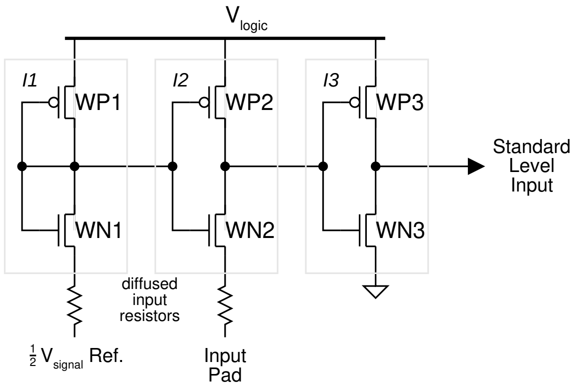


Shown above are there possibilities for the voltage waveform at the source of a series-terminated transmission line. At the top, we have a matched impedance situation. The middle diagram shows a case where the series terminating impedance is too large, and the bottom diagram shows a case where the series impedance is too small. *N.B.* T on the time axis is the unidirectional transit time, t_{pd} from Equation 3.

Figure 2: Series-Terminated Source Transitions

3.2 Receiver

The receiver must convert the low-voltage swing input signal to a full-swing logic signal for use inside the component. In the interest of high-speed switching, we want a receiver which has high gain for small signal deviations around the mid-point between the signaling supplies. [2] and [9] introduce suitable differential receivers. Figure 4 shows one such receiver.



Shown above is a receiver after [9]. Inverters $I1$ and $I2$ are identical devices ($WP1 = WP2$, $WN1 = WN2$). $I1$ biases $I2$ into its high-gain region. When the voltage on the input pad is slightly higher or lower than the half signaling voltage reference, $I2$ amplifies the voltage. $I3$ standardizes the output of $I2$ for use inside the component. $I3$ should be sized to have the same voltage midpoint as $I1$ and $I2$.

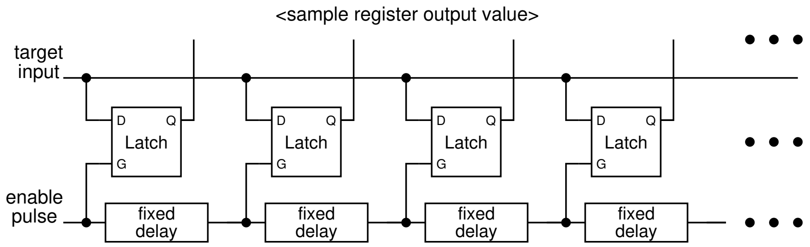
Figure 4: CMOS Low-voltage Differential Receiver Circuitry

4 Sample Register

This section introduces the *sample register*, the key circuitry for timing extraction. A sample register is a string of latches enabled at closely spaced time intervals. Each latch "samples" the binary value of the signal under test during the time it is enabled. By rippling the latch enables in rapid succession, the sample register captures a discrete representation of the time behavior of a signal. In this section, we develop sample register circuits starting from an "ideal" model and progressively refining the circuitry into practical implementations.

Alternately, one could consider delaying the target input signal as seen by each latch and using a single, register-wide enable rather than delaying the enables. We focus our discussion around a delayed enable since we generally have more control over the timing of the enable pulses we generate than we do over a random signal whose timing we wish to capture.

The sample register is compatible with scan-based Test-Access Ports (TAPs), such as the JTAG/IEEE 1149 standard [3] TAP. The TAP can be used to initiate events which the sample register captures and to offload the data captured in the sample register.



An ideal sample register is composed of a sequence of latches each enabled at fixed delays. When a timing event occurs, a short enable pulse is driven into the sample register. Each sample latch records the value seen by the receiver when it was last enabled. After the enable pulse propagates through the sample register, the sample register holds a discrete-time sample of the target input.

Figure 5: Ideal Sample Register

4.1 Ideal

An ideal sample register would consist of an infinitely long string of latches, each enabled at uniform time intervals (See Figure 5). When the enable pulse fires at the beginning of the delay chain, the sample register captures a discrete-time representation of the attached input signal. We seek to approximate the ideal behavior with a reasonably small, finite-length sample register and employ simple circuitry to generate the sequence of enables necessary to capture timing samples.

4.2 Inverter Timing Chain

For many applications, a pair of inverters will suffice to form the inter-sample-bit delay (See Figure 6). The delay through a single inverter is often the finest granularity of timing used in a design. With a little care during layout, the geometry and loading of each inverter in the delay chain can be made identical. Consequently, the only variation between delays will be due to process variation across the die. Since a single sample register will typically occupy only a small region of the die, the processing variation among inverters in a sample register will be minimal.

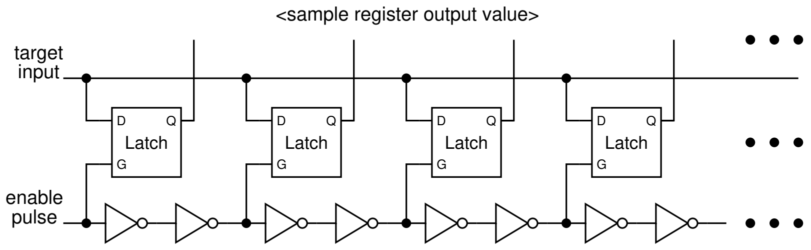
4.3 Sliding Window

If we can cause the event we are timing to occur under scan control, we can tradeoff time for space. We do not have to capture the entire waveform in a single event. We can reuse a small sample register in time to capture a long waveform. The temporal placement of the sample register can be controlled via the TAP, and the composite waveform can be reconstructed off chip. Figure 7 shows the basic sliding window concept.

Figure 8 shows one possible implementation for the sliding window. The sample pulse is recycled after rippling through several sample delays. A scan-loaded configuration is compared against a trip count to allow the sample pulse to be recycled for a predetermined number of times. After the enable pulse settles following a trigger event, the sample register will contain the values corresponding to the last time the pulse was allowed to ripple through the delay chain.

Note that we depict the ripple pulse recycling *before* the end of the sample register. It is unlikely that the delay on the recycle path can be accurately matched to the inter-sample time defined by the inverters. By recycling the ripple from a point prior to the end of the sample register we can do two things: (1) provide overlap between sample windows and (2) make sure that there is always an inverter-pair delay between adjacent samples used to reconstruct the longer waveform. As we will see in the following section, with sufficient overlap, calibration can help us factor out any delay anomalies associated with the recycle path.

The choice of how many bits to include in the sample register and recycle path will depend on the relative speed of operation of various logic functions in the target technology. In particular, the operational frequency of the counter-comparator combination will set a lower limit on the delay between successive ripple pulses. For example, a technology with 100 ps minimum inverter delays and a maximum counter operational frequency of 500 MHz



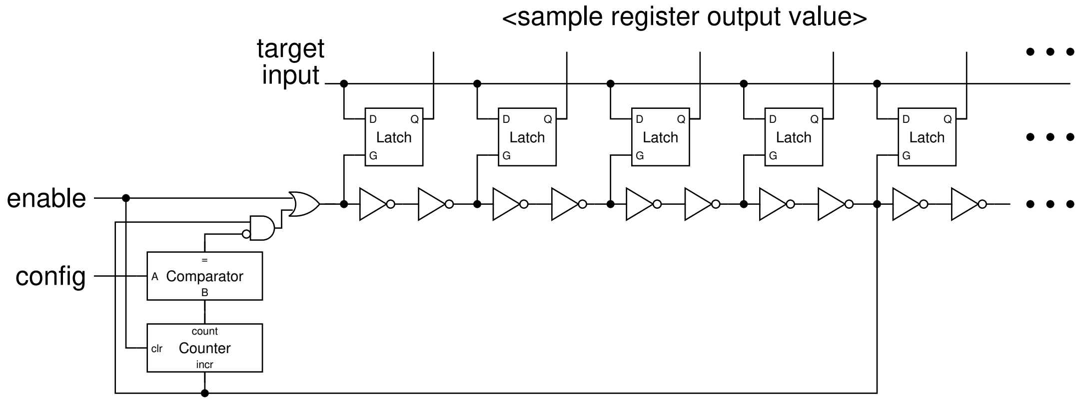
A simple sample register can be implemented using a pair of inverters to provide the fixed, inter-sample-bit delays.

Figure 6: Sample Register

Target Data	000000000111111111...
Sample 1	000000000111111111...
Sample 2	000000000111111111...
Sample 3	000000000111111111...

Using a small, fixed-size, sample register, we can capture a portion of the discrete-time waveform for a signal with each timing event. If we vary the placement of the capture window and repeatedly fire the timing event, we can capture the entire waveform over a series of such samples.

Figure 7: Sliding Window



We can implement the sliding window in our sample register by recycling the sample-enable pulse a configurable number of times. The values left in the sample register after a timing event will correspond to the data capture during the last cycle made by the enable pulse.

Figure 8: Recycling Sample Register

would require a minimum of 10 inverter pairs in the recycling portion of the sample delay chain.

With slightly lower accuracy, it is possible to use only a single latch in the sample register. As shown in Figure 9, a mux can be used to select the fine timing delays while the counter-comparator combination selects the coarse-grain timing window. The delay between sample bits in this scheme will not generally be as accurate as the earlier circuits, but may be sufficient for many applications.

4.4 Calibration and Sharing

With the circuits shown so far, we only know when the signal is occurring in units of inverter-pair delays. Since process and environmental variation can easily account for a factor of two variance in an inverter delay, inverter-pair delays alone are not sufficient to extract fine-grained timing information. If a known timing source, such as the component clock, is available we can mux the sample input between the sample register and the known timing source to calibrate the inter-sample-bit delay time (See Figure 10). A known-frequency clock will allow us to determine the timing of events on the sample register and reassemble the overlapped, sliding windows appropriately (See Appendix A).

The mux can also be used to share a single sample register between several target signals. This muxing can be used simply to minimize the need for sample registers. It can also serve to acquire accurate, relative timing information for groups of related signals.

For example, one might share a single sample register among the bits of an 8-bit data bus and its data strobe. This arrangement would provide accurate timing information on the relative occurrence of data bit transitions to each other, as well as, providing an indication of when data bit transitions occur in relation to the data strobe.

4.5 Tighter Timing

In applications where timing accuracy tighter than a pair of inverter delays is required, a fine-resolution, variable-delay buffer can be placed at the front-end of the delay path (See Figure 11). Some variable delay buffers developed for CMOS, Phase-Locked Loop (PLL) circuitry are suitable for this application. Figure 12 shows a voltage-controlled, variable-delay buffer which operates by varying the capacitive load seen by each stage. Horowitz [7] details a phase interpolator which smoothly varies the phase in 15 steps between two references under digital control. Such a phase interpolator can be applied to an inverter-pair delay to provide a resolution of roughly one-eighth of an inverter delay.

4.6 Enable Pulse

To acquire a "sample", we must be able to initiate both the event under test and the sample-enable pulse. When we are using a scan-based TAP for offloading the sample registers, it will be most convenient to initiate these events under scan control. In practice, we generally want the enable pulse to fire synchronized to the event we wish to observe on the IC. Scan control is used to prime the enable pulse to trigger on the next synchronization event and to initiate the event for observation. For example, when the component is in a scan testing mode, the standard scan register load facilities can be used to cause signal transitions within the IC or at the IC boundary. Once fired, circuitry inhibits the enable pulse from firing again until we have had a chance to offload the sample register and configure it to capture the next timing event.

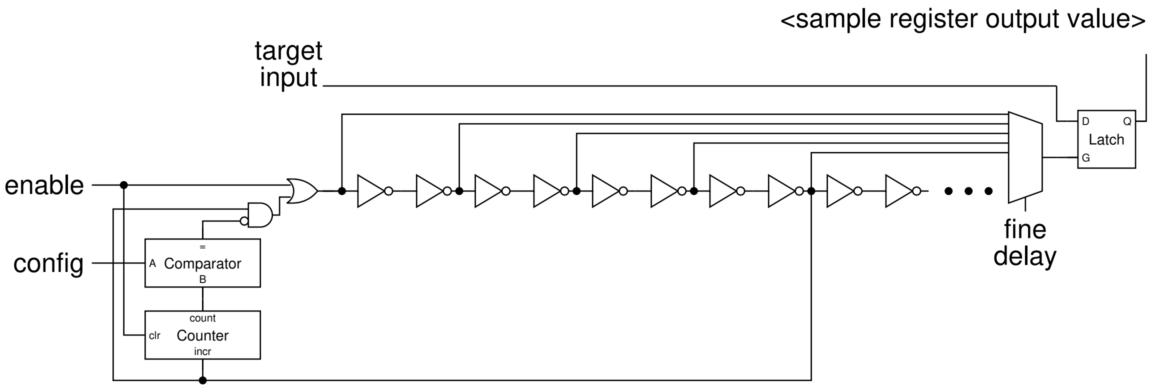
4.7 Summary

Combining these techniques we can repeatedly fire an event we wish to time and sample its behavior in narrow, fixed-size windows. By integrating the information acquired across multiple samples at varying window offsets and calibrating to known frequency and phase sources, we can build up an accurate, discrete-time representation of a signal on the IC. We can easily achieve timing resolutions down to two inverter delays and, with care, can achieve even tighter resolutions. Data capture and acquisition can be completely controlled through a scan-based TAP.

5 Impedance Matching

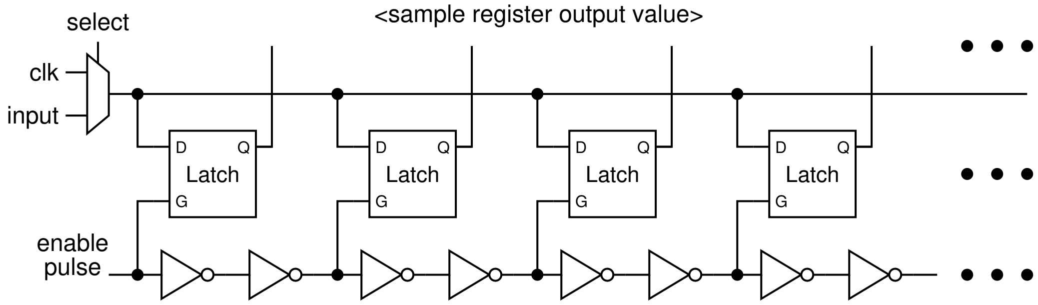
Each controlled impedance pad is constructed from:

1 Driver (Figure 3)
2 Receiver biased to trip half way between the high and low signaling supplies (Figure 4)
3 TAP scan register as shown in Figure 13



With slightly less accuracy, a fine-grained, variable-delay buffer can be employed reducing the number of sample latches required to one.

Figure 9: Sample Register with Single Latch



Target Data	000000000111111111...
Clock Signal	000011110000111100...
Data Sample 1	000000000111111111...
Clock Sample 1	000011110000111100...
Data Sample 2	000000000111111111...
Clock Sample 2	000011110000111100...

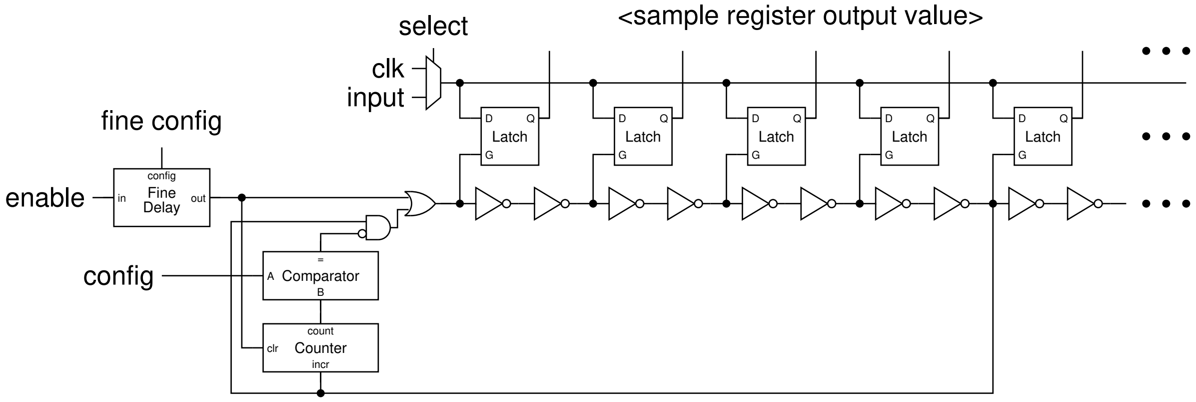
To calibrate the sample register, we can mux in a known frequency and phase reference such as the system clock.

Figure 10: Calibrated Sample Register

The receiver plays the dual purpose of (1) bringing the signal onto the chip when the pad is acting as an input and (2) monitoring the source of the transmission line when calibrating driver impedance. The pad's sample register is enabled whenever the core logic toggles the value driven into the output pad. Following such transitions, the sample register records the value on the output pad as seen by the receiver.

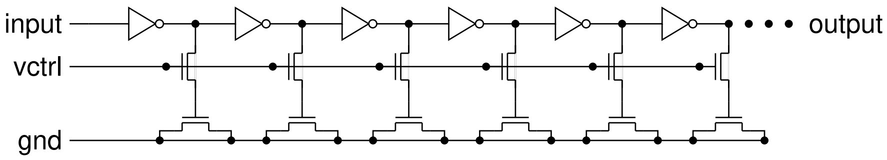
Since the receiver is biased to trip at the midpoint voltage, we can use the value recorded in the sample register to determine when the series-terminated transmission line is matched. When the

VLSI Design, Vol. 7, No. 1 (1998) 111-129



We can achieve higher resolution by adding a fine-grained, variable-delay buffer in the enable path preceding the inverter chain. This allows us to vary the timing of the sample taken by sub-inverter-pair quanta.

Figure 11: Sample Register with Fine Prefix Delay



Shown here is a voltage-controlled delay line (VCDL) after [1] and [8]. Varying VCTRL effectively controls the amount of capacitive load seen by the output of each inverter stage and hence the delay through each inverter stage. The number of stages one uses in the VCDL will depend on the range of delays required from the buffer.

Figure 12: Voltage-Controlled Variable-Delay Buffer

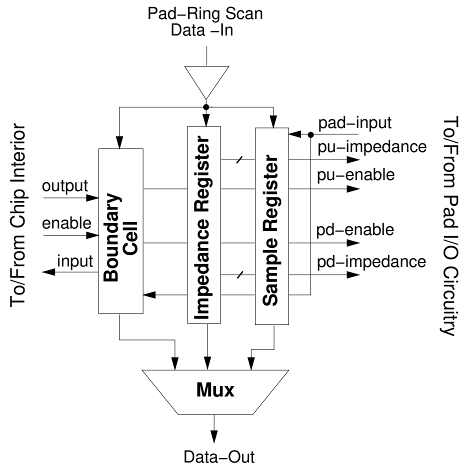
drive impedance is too low, the driver will quickly drive the line past the half-way point and the receiver will capture the transition. If the impedance is too high, the driver will not drive the voltage past the half-way point and the receiver will not see a transition until subsequent reflections bring the voltage past the midpoint. By firing a series of test transition and recovering the sampled result, an off-chip controller can select an impedance setting where the drive impedance is well matched to the transmission line impedance.

The impedance selection time is almost entirely dictated by the bandwidth to the off-chip controller. Impedance setting using a scan-based TAP can take half a millisecond for a single pin. For an entire chip with hundreds of pins, scan-based impedance setting can take on the order of 100 ms (See Appendix B).

Figure 14 shows data collected from a test chip (See Section 9) while scanning through impedance settings. Figure 15 shows both ends of a series-terminated transmission line after the driver has been automatically matched to the line impedance using the data collected in Figure 14.

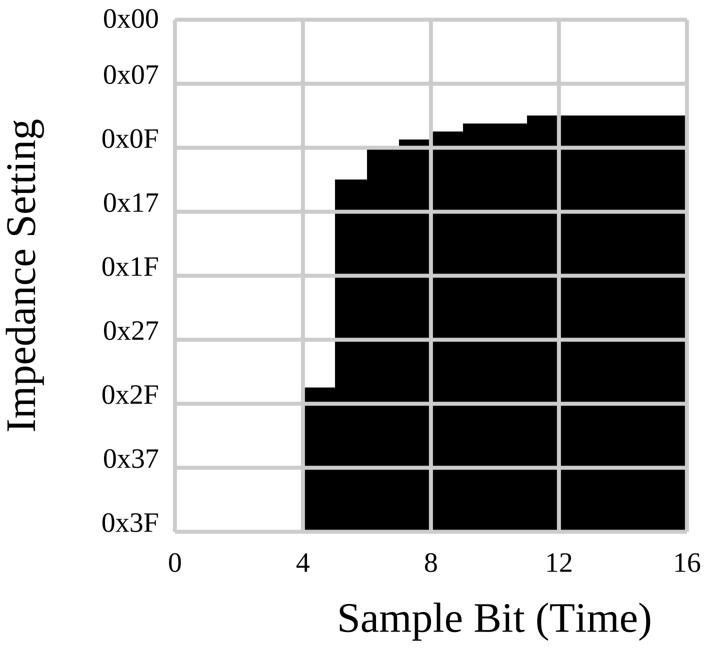
The sample register is important to this application for two key reasons:

1. The delays through the driver and receiver will depend on IC processing. By capturing a window of the signal, we can be sure to capture the transition regardless of processing.
2. When we cannot control the length of the attached transmission line, it is difficult to know when the transmission line voltage corresponds to the initial drive or reflections. Coupled with process variation, we cannot sam-



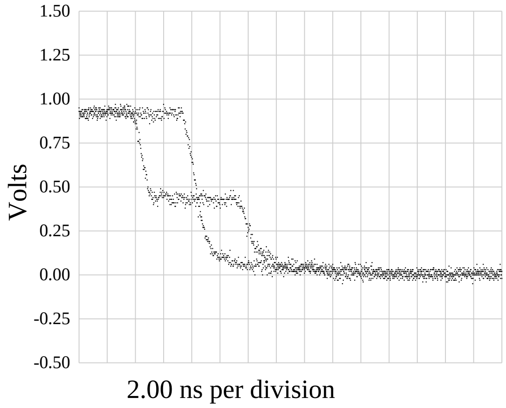
"Boundary Cell" contains the standard boundary scan registers for a bidirectional i/o pad. "Impedance Register" holds the digital value which controls the pad drive impedance. "Sample Register" is a sample register as described in Section 4.

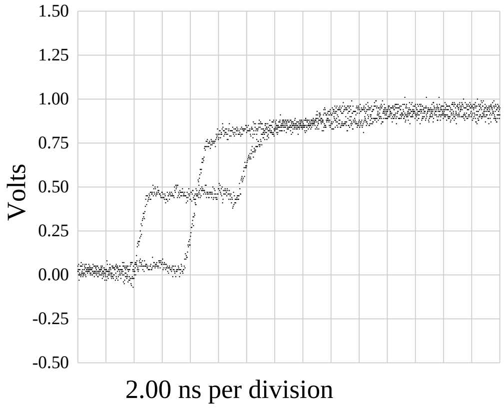
Figure 13: Bidirectional, Controlled-Impedance Pad Scan Architecture



The data above accompanies a low-to-high transition of the output value. The dark areas indicate that the receiver saw a high value, while the light areas indicate a low value. Impedance setting 0x3F corresponds to all impedance transistors enabled which is the lowest impedance setting while 0x00 corresponds to all impedance transistors disabled, the highest setting.

Figure 14: Sample Register Data Following low-to-high Transition of Output Value.





Scope traces show waveforms near both ends of 50Ω PCB trace.

Figure 15: Matched Impedance Transitions using Sample Register Data

ple at any single point in time and know we are looking at the steady-state line voltage between initial drive and first reflection.

The sample register allows us to see transitions occur. The transitions act as calibration marks, informing us when various events occur.

In effect, we have built a poor Time-Domain Reflectometer (TDR) which we use to match the driver impedance to the line impedance. The discrete-time sample in the sample register is coarser than real TDRs, the rise time on the signals is much slower, and the length of the line monitored is limited by the window size captured by the sample register. For comparison, we connected one of the test pads to a small wire terminating in a short-circuit and recorded the sample data for various impedance settings. Figure 16 compares the sample data to a real TDR waveform. The time range of the test pad is limited because we used a 16-bit sample register without the recycling technique (See Section 9). The sliding window (Section 4.3) allows us to extend the time range captured considerably for modest additional silicon real-estate.

Section 9 summarizes the characteristics of the test pad used to acquire the data shown in Figures 14, 15, and 16. Section 10 elaborates on the limitations of this technique as well as the expected usage patterns.

6 Delay Adjustment

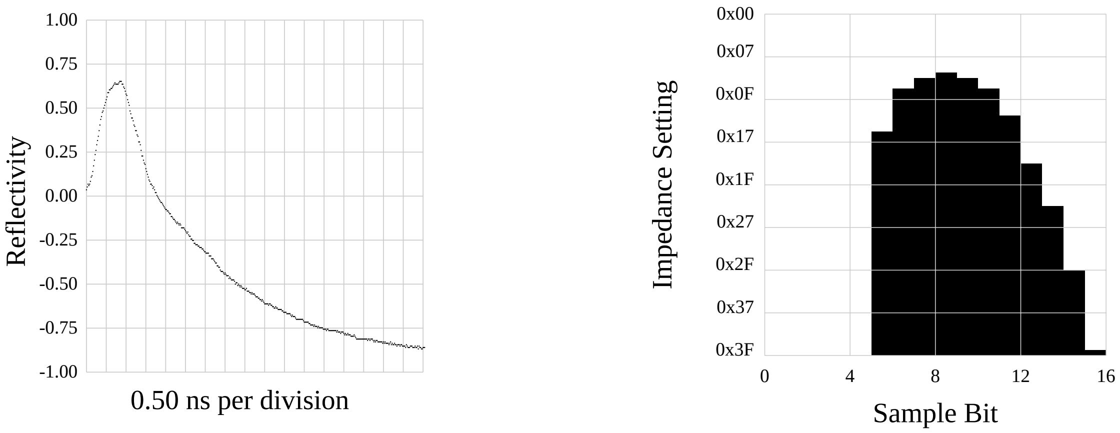
6.1 Mechanism

We can use similar techniques to adjust the timing of key IC signals. Figure 17 shows a variable-delay buffer suitable for coarse-grain delay adjustment. Since this buffer uses inverter pairs as the basic, unit-delay element, it provides the same granularity of adjustment as most of the sample register designs presented in Section 4. For finer grained delay adjustments, we can borrow variable-delay elements from PLL circuits such as the VCDL buffer (Figure 12) or Horowitz's phase interpolator mentioned in Section 4.5.

6.2 Comparison with PLLs

Phase-Locked Loops are commonly employed to match timing of on-chip clock signals to external references. In such cases, where the signal is periodic with fixed frequency, on-chip circuitry can close the feedback loop to adapt component timing to match system timing. However, PLL techniques cannot be applied to non-periodic control signals and data paths. Further, traditional PLLs cannot be used to guarantee the simultaneous arrival of the bits of a wide data bus.

Our TAP-based timing extraction and timing control can provide in-system adjustment of on-chip



Left – TDR Profile for wire with a short-circuit. Right – Sampled data for same wire.

Figure 16: Comparison of TDR and Sample Register Data

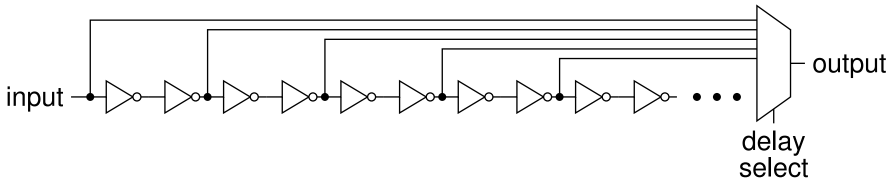


Figure 17: Multiplexor-Based Variable-Delay Buffer

timing for these non-periodic signals. The TAP can be employed to force events to occur and capture their timing relationships. Through the TAP, we can adjust the delay controls to servo the on-chip delays until the proper timing relationships are achieved. The feedback loop using TAP-based timing extraction and control is, of course, much slower than the feedback loops in conventional PLLs and does not operate continuously. Coupled with generally coarser-grained timing information, this makes TAP-based timing control unsuitable for the fine-grained timing adjustment provided by competent PLLs in the same technology. The coarser control of TAP-based timing does bring many of the advantages of feedback control to non-periodic signals and signal groups.

6.3 In-system Tuning

On-chip delay adjustment allows us to tune the timing of events to the target system. Once a component is deployed in its final system many of the variables which had to be considered during design are fixed and will remain effectively constant during operational epochs. Such variables include:

- IC processing of *all* ICs in the target system, including this one
- External interconnect characteristics (*e.g.* path length, line impedance, propagation delay, capacitive loading)
- Target system clock frequency

Other variables may vary during operational epochs, but do so relatively slowly (*e.g.* component temperature and operating voltage). If we can monitor changes in these parameters (*e.g.* with on-chip temperature sensors) we can often treat these parameters as constants, retuning whenever significant environmental changes make retuning necessary. Once system delays are fixed, and can be measured using our timing extraction techniques, in-system component operation can be specialized to these system characteristics. By specializing component timings around system characteristics, we can achieve higher performance than is possible when our design must allow for all possible variations in system parameters. In-system delay adjustment effectively gives us most of the advantages of self-timed logic without incurring the complexity and testability problems associated with asynchronous logic.

7 Transmission Line Timing Adjustment

For high-bandwidth signaling over long transmission lines, we can pipeline multiple data bits on the transmission line. This wire pipelining requires:

1. We know how many clock cycles it takes to traverse each transmission line interconnect.
2. We guarantee that data transitions do not occur during the setup to hold time window of the receiving IC.

Computers such as the Cray-1 [10] and CM-5 [13] satisfy these criteria by carefully selecting the interconnect cable lengths and designing the basic system around the logical lengths of each interconnect. Using the techniques we have introduced here, it is possible to satisfy these two conditions by monitoring the transmission line reflections and adapting the output timing to the length of the connected transmission line.

To handle long transmission lines, we add a tunable delay and sliding-window sample register with a muxed clock for calibration to the pad design described in Section 5. We can tune the output impedance to the transmission line as described previously. However, while scanning impedances the longer effective sample window gives us an additional piece of information, the timing of the first reflection arrival. When the transmission line impedance is set slightly above the transition point, the receiver will trip when the first reflection arrives.[1] Scanning through impedance settings thus tells us both when the source is driven and when the first reflection occurs. Assuming synchronous clock distribution and symmetry of transit times across the transmission line, this also allows us to determine when the signal arrives at the destination end of the transmission line. We can take this time and determine (1) how many clock cycles it requires to traverse this interconnect and (2) where during the clock cycle the signal is arriving at the far end of the transmission line. We can then tune the variable delay associated with the output signal so that the transition at the destination is guaranteed to occur outside of the setup to hold time window around the clock after taking into account any necessary uncertainties associated with the delay through the input receiver.

By combining the sample register with series-terminated, transmission-line signaling, we can tune the arrival of a transition at the far end of a variable length interconnect. This kind of tuning is not possible with conventional PLL circuits. Figure 18 shows a suitable pad scan architecture including the tunable output delay.

8 Implications on Interconnect Testing

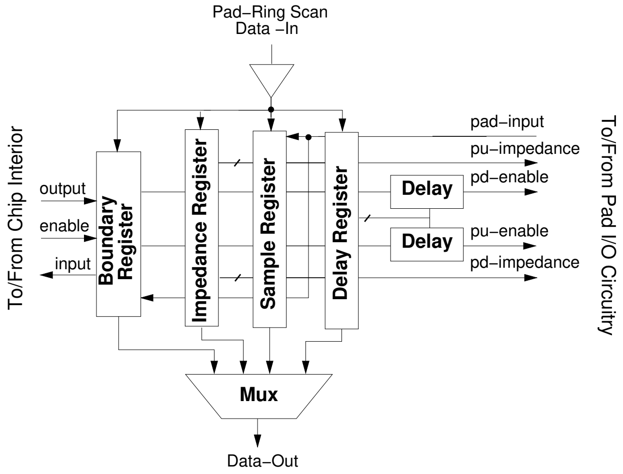
It is worthwhile to note that techniques presented here allow us to test the dynamic properties of our interconnect media. Today, TAPs are commonly employed to test out the DC characteristics of ICs and interconnect integrity. Standard TAP techniques, however, cannot identify interconnect faults which only affect high-speed signals. For example, conventional TAP interconnect testing cannot identify the impedance discontinuity arising from a poorly seated connector or a short to some piece of foreign material. With the pad architecture presented here, the TAP can capture the dynamic profile of the voltage waveform resulting from signaling events. This allows the recovery of TDR-like data for the attached interconnect. Consequently, these techniques allow us to extend our TAP testing to identify the interconnect faults which affect high-speed signals.

9 Implementation

We have implemented a prototype, matched impedance i/o pad which incorporates many of the techniques described here [5]. Table 19 summarizes the key characteristics of the prototype pad and Figure 20 shows the layout for the bidirectional i/o pad. This pad's scan architecture matches the one depicted in Figure 13. Note that the sample register occupies 350μ of the test pad length. In application, one could construct a single sample-register

[1] Technically, as long as the impedance is between Z_0 and $4Z_0$ the initial drive will not trip the receiver, but the reflection will.

VLSI Design, Vol. 7, No. 1 (1998) 111-129



Shown above is the revised bidirectional pad architecture incorporating variable-delay buffers in the output path as well as a scannable register to control the delay buffers.

Figure 18: Adjustable Delay, Bidirectional Pad Scan Architecture

Process	0.8μ CMOS (HP)
Latency	
(driver)	2 ns
(receiver)	1.5 ns
Register Bits	
(impedance control)	6 per direction
(sample register)	16
Impedance Range	40 to 100 Ohms
Area	$150\mu \times 930\mu$
Power	10mW+2mW/100MHz

Figure 19: Prototype Matched Impedance I/O Pad Characteristics

with muxed inputs from many adjacent pads. In this way the area cost of the sample-register could be amortized across multiple pads as suggested in Section 4.4.

For testing purposes, we drove the test component's TAP from the parallel port of a Personal Computer (PC). Software running on the PC would:

1 load impedance configurations
2 create output transitions using the component's boundary scan cells
3 offload sample results
4 search through the sequence of recovered sample results similar to that shown in Figure 14 to select an impedance setting
5 install the selected setting in the impedance configuration and return the component to its normal, non-scan, operating mode

The data shown in Figures 14, 15, and 16 came from this test component using the PC for off-chip control.

In a deployed application, the off-chip control task would be handled by the same embedded controller which managed the Test-Access Port and device testing. Standard components, such as National Semicondutor's SCANPSC100F [11] or Texas Instrument's SN74ACT8990 [12], are now available for interfacing microprocessors to IEEE 1149.1 compatible scan chains.

10 Limitations

In this section we review some of the costs and limitations of these on-chip sensing and adjustment techniques. We briefly address the impact of these limitations on the pragmatic application of these techniques.

Point-to-Point Signaling As noted, the specific technique described here is primarily applicable to single driver, single receiver, series-terminated signaling applications. The tuning behavior depends on the reflection profile of the series terminated transmission line for proper operation.

Single Impedance Media The techniques described here assume the interconnection media,

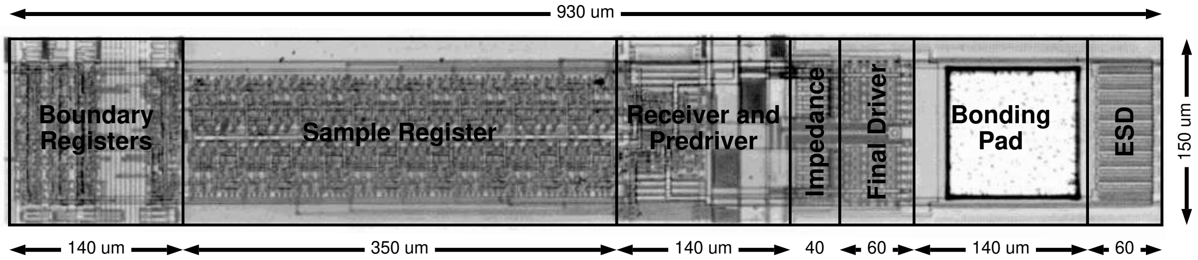


Figure 20: Prototype Matched Impedance I/O Pad

while varying in impedance, is characterized by a single, homogeneous impedance between the source and the driver. This is typically the case if the interconnect is a printed-circuit board or cable between ICs. However, if the ICs are connected through multiple media this might not be the case. For instance, if two ICs are communicating over a long cable and each IC has a long wire run between the cable and the IC on its attached printed circuit board, the intervening interconnect could be characterized by three distinct impedance regions. The techniques presented here will allow one to identify the impedance discontinuities, but not to compensate for them.

Of course, one could use the techniques presented here to build an impedance matching buffer component to place at each potential impedance discontinuity. The impedance matching buffer could then separately match to each interconnect segment. Such a scheme would, however, add i/o delay to the signaling path for each such impedance matching buffer encountered.

Area This technique does require dedicated, on-chip silicon area. As noted in the previous section, a 16-bit sample register occupied just under $350\mu \times 150\mu$ [5]. The prototype sample register included the inverter chain, sample latch, shift register, and a 16-bit configuration register, but did not include any recirculation or calibration circuitry. For comparison, the standard bidirectional i/o pad boundary-scan registers in the same design occupied $140\mu \times 150\mu$. Layout for the standard boundary-scan i/o registers was partially determined by control signal routing, while the the sample register contains local connections and is dominated more by the size of shift and configuration registers. Using the recirculation techniques suggested in Section 4, one could build a smaller, 8- to 10-bit sample register and then build a 4- to 5-bit counter and comparator in comparable space. Calibration support then requires the addition of an input mux along with an attached configuration register.

Recall from Section 4.4 that the input mux can be expanded in order to share a single scan register among several signals. For bussed signals, a single sample register would typically be shared among a series of 4 to 8 adjacent lines to amortize the area cost required.

Configuration and Tuning Latency Up front tuning latency using the sample register scheme can be moderately large.

- Sample registers can collect many bits of data per timed signal per experiment, but must still offload such data via the low bandwidth, serial scan interface.
- To keep the area requirements down, recirculating and shared sample registers reuse the sample register in time. Consequently, many timing experiments must be performed in sequence in order to reconstruct a single waveform.

These effects make TAP-based timing extraction and configuration a moderately high-latency operation. For example, a 160-pin component using the prototype pad from Section 9 requires 50-60 ms to tune all pads. Appendix B shows how to estimate configuration latencies based on scan and component architecture.

In practice tuning would occur initially at system startup time and thereafter only when environmental characteristics change. As long as the environmental characteristics change slowly, the tuning latency does not have an adverse effect on signaling operation.

Periodic Retiming Requirements As noted in Section 6.3, tuned parameters such as delay or impedance will depend on some slowly-changing environmental characteristics such as temperature and attached hardware configuration. These parameters will need to be retuned whenever environmental characteristics drift significantly from the point of tuning. The way this retuning fits into system operation will vary considerably among applications. In systems with adequate error detection, the detuning can be recognized by the error detection mechanism, and retuning may serve as a primary responses to excessive errors. In systems without this kind of error detection, more preemptive measures may be required. For example, a crude, on-chip temperature sensor can serve as an early warning indicator so that retuning can compensate for changes in temperature.

Off-Chip Controller These techniques will require a moderately complex off-chip controller for waveform extraction and reintegration. A personal computer or low-end workstation is both sufficient and economical for in-system testing and tuning. For impedance and delay tuning applications, the controller should be an inseparable part of the base system. In many systems, the task can be assumed by an existing processor in the system. Some systems may require an additional, embedded microcontroller to orchestrate tuning and configuration functions.

11 Summary

By exploiting the information available at the source end of a series-terminated transmission line, we can identify important characteristics of our interconnect. Employing a TAP accessible sample register, timing control, and impedance control, we can match a series-terminated transmission-line driver to its attached transmission-line, in system. Specifically, we take discrete-time, binary samples of the voltage seen by the driving pad at various impedance settings. Using the sample feedback, we can determine both:

1. the driver impedance setting which best matches to the transmission line impedance
2. the arrival time of the signal at the far end of the transmission line

In effect, we get the capabilities of a crude, on-chip TDR. Using this information and fine-grained timing adjustment at the source end of the transmission line, we can reliably pipeline the transmission of data bits over variable length interconnect media. This pipelining allows high bandwidth signaling even when interconnect distances are long.

In general, these techniques allow us to factor out process variation for the ICs and interconnect media and adapt to system specific parameters such as interconnect impedance and length. Additionally, these techniques allow us to use a TAP to determine the integrity of our interconnect for high-speed signaling.

References

[1] M. Bazes. A Novel Percision MOS Synchronous Delay Line. *IEEE Journal of Solid-State Circuits*, 20(6):1265–1271, December 1985.

[2] Barbara A. Chappell, Terry I. Chappell, Stanley E. Shuster, Hermann M. Segmuller, James W. Allan, Robert L. Franch, and Phillip J. Restle. Fast CMOS ECL Receivers With 100-mV Worst-Case Sensitivity. *IEEE Journal of Solid-State Circuits*, 23(1):59–67, February 1988.

[3] IEEE Standards Committee. *IEEE Standard Test Access Port and Boundary-Scan Architecture.* IEEE, 345 East 47th Street, New York, NY 10017-2394, July 1990. IEEE Std 1149.1-1990.

[4] André DeHon. Robust, High-Speed Network Design for Large-Scale Multiprocessing. AI Technical Report 1445, MIT Artificial Intelligence Laboratory, 545 Technology Sq., Cambridge, MA 02139, February 1993.

[5] André DeHon, Thomas F. Knight Jr., and Thomas Simon. Automatic Impedance Control. In *ISSCC Digest of Technical Papers*, pages 164–165. IEEE, February 1993.

[6] Thaddeus J. Gabara and Scott C. Knauer. Digitally Adjustable Resistors in CMOS for High-Performance Applications. *IEEE Journal of Solid-State Circuits*, 27(8):1176–1185, August 1992.

[7] Mark Horowitz, Andy Chan, Joe Cobrunson, Jim Gasbarro, Thomas Lee, Wing Leung, Wayne Richardson, Tim Thrush, and Yasuhiro Fujii. PLL Design for a 500MB/s Interface. In *ISSCC Digest of Technical Papers*, pages 160–161. IEEE, February 1993.

[8] Mark G. Johnson. A Variable Delay Line PLL for CPU-Coprocessor Synchronization. *IEEE Journal of Solid-State Circuits*, 23(5):1218–1223, October 1988.

[9] Thomas F. Knight Jr. and Alexander Krymm. A Self-Terminating Low-Voltage Swing CMOS Output Driver. *IEEE Journal of Solid-State Circuits*, 23(2), April 1988.

[10] James S. Kolodzey. CRAY-1 Computer Technology. *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, CHMT-4(2):181–186, June 1981.

[11] National Semiconductor Corporation, 2900 Semiconductor Drive, P.O. Box 58090, Santa Clara, California 95052-8090. *SCAN Databook*, 1993.

[12] Texas Instruments, P.O. Box 655303, Dallas, Texas 75265. *SCOPE Product Information*, 1992.

[13] Thinking Machines Corporation, Cambridge, MA. *CM5 Technical Summary*, October 1991.

A Calibration

Sample register calibration is required for two reasons.

1. The time delay between sample bits varies with process variation.
2. The timing on the recycle path in the sliding window also varies and is generally different from the time delay between sample bits.

As a result, when we recover a pair of adjacent samples, we do not immediately know the amount of overlap between samples. Figure 21 depicts an example of the alignment problem. As suggested in Section 4.4, if we can mux a known frequency source into the sample register, we can calibrate both bit times and sample overlap. This in turn allows us reconstruct composite waveforms from a collection of window samples and to determine coarse-grain, absolute timing between signals.

Fast Clock Case If the sample register is long enough, relative to the clock frequency, we can capture an entire clock phase in one sample register. Once captured, we can divide the clock phase time by the on, or off, duration, in bits, to get the inter-sample-bit time. That is

$$\begin{aligned} t_{sbit} &= \frac{t_{clk_{high}}}{n_{high-samples}} \\ t_{sbit} &= \frac{t_{clk_{low}}}{n_{low-samples}} \end{aligned} \tag{6}$$

Reassembling windows is easy in this case since we immediately discover the inter-sample-bit time and can align corresponding edges between samples (See Figure 22).

To be in the fast clock case, we need:

$$n_{sbits} \geq 2 + \left\lceil \frac{\min(t_{clk_{low}}, t_{clk_{high}})}{t_{sbit}} \right\rceil \tag{7}$$

For a 500 MHz clock with a 50% duty cycle in a technology with a minimum inverter delay of 100 ps this means:

$$n_{sbits} \geq \left(2 + \left\lceil \frac{1 \text{ ns}}{100 \text{ ps}} \right\rceil = 12 \right) \tag{8}$$

Slow Clock Case When the available calibration clock is slower it may not make sense to build a sample register long enough to cover half a clock period. In this mode we have to look for two pieces of information independently:

- T_{scycle} – the time between successive placements of the sample register window
- t_{sbit} – the time between successive bits within the sample register

Each edge occurs at some offset within a sample register (n_{bp}) and at some cycle offset (n_{wp}). Given a pair of edges separated by time T_{e-e}:

$$(n_{wp_2} - n_{wp_1}) T_{scycle} + (n_{bp_2} - n_{bp_1}) t_{sbit} = T_{e-e} \tag{9}$$

From here there are two ways we can solve for t_{sbit} and T_{scycle}.

1. If we can see an edge in two successive windows, we know:
$$T_{scycle} + n_{bp_2} \cdot t_{sbit} = n_{bp_1} \cdot t_{sbit} \tag{10}$$
The only time when this will never occur is when there is no overlap. That is:
$$n_{sbits} < \frac{T_{scycle}}{t_{sbit}} + 1 \tag{11}$$
See Figure 23 for an example of this case.
2. We can also derive a relationship between t_{sbit} and T_{scycle}, if we can get two different sets of $((n_{wp_2} - n_{wp_1}), (n_{bp_2} - n_{bp_1}))$ pairs. An example of such differing pairs is shown in Figure 24.

Either of these cases provide a second equation in two unknowns allowing us to solve for the bit and cycle times and calibrate the sample windows.

Alignment 1	$\boxed{00001111}$
	$\quad\quad\;\;\boxed{11111111}$
Alignment 2	$\boxed{00001111}$
	$\quad\quad\quad\boxed{11111111}$
Alignment 3	$\boxed{00001111}$
	$\quad\quad\quad\;\boxed{11111111}$
Alignment 4	$\boxed{00001111}$
	$\quad\quad\quad\quad\boxed{11111111}$

Given a pair of samples, 00001111 and 11111111, we do not immediately know how the sample bits overlap between samples. Shown above is the basic range of possibilities for this pair of samples assuming that the sample register has been designed to guarantee at least one bit of overlap between a pair of adjacent sample windows. Sub bit-time shifts are also possible.

Figure 21: Example of Alignment Problem

Clock Signal	000011110000111100001111...
Clock Sample 1	0$\boxed{0001111}$0000111100001111...
Clock Sample 2	000011$\boxed{1100001}$11100001111...
Clock Sample 3	00001111000$\boxed{0111100}$001111...

When the calibration clock is fast relative to the length of the sample register, we can capture an entire clock phase and directly determine the inter-sample-bit time. Shown here is a 7-bit sample register with 2 bits of overlap per window. The calibration clock has a period of 8 sample bit delays and a 50% duty cycle.

Figure 22: Fast Clock Sample Register Calibration

If T_{e-e} is not an exact multiple of T_{scycle}, the edge positions within the sample windows will drift from window to window. This will guarantee that one of the above two cases will occur allowing calibration. If T_{e-e} is an exact multiple of T_{scycle}, $(n_{bp_2} - n_{bp_1}) = 0$, so we immediately know T_{scycle} but do not know t_{sbit}. There is also a potential problem if T_{e-e} is an exact multiple of t_{sbit} for certain values of the multiplier. If we have no control over the calibration clock edges and wish to avoid these exceptional cases, it will be necessary to make the timing on the recycle path adjustable. For example, an optional, extra inverter pair delay in the recycle path would allow us to change T_{scycle} so that it was no longer a proper divisor of T_{e-e}. Of course, if T_{scycle} is close to being a multiple of T_{e-e} it may take many calibration clock periods, and hence windows, to guarantee the adequate positional shift to guarantee sufficient calibration data. The optional recycle delay can also be useful in this case to reduce the required time coverage for the sliding window sample register.

B Impedance Tuning Time

The most straightforward way to set the impedance involves a sequence of:

1 start with minimum impedance
2 load in current impedance value
3 force a transition
4 offload resulting sample register
5 increment current impedance value and repeat at step 2 until all impedance values have been tested

Once this sequence of data has been collected, the controller has a collection of information like that shown in Figure 14. The impedance setting can then be found from this data (See [4] for additional details). In practice, it is necessary to set the pull-up and pull-down impedance separately and a few iterations are required to converge. The subsequent iterations can almost certainly be done without scanning through the full range of possible

Clock Signal	000011111111111100000000...
Clock Sample 1	00001$\boxed{11111111}$11100000000...
Clock Sample 2	0000111111$\boxed{11111100}$000000...
Clock Sample 3	000011111111111$\boxed{10000000}$0...

Shown here is a slow clock case where a pair of calibration clock edges never occur in a single sample window. When we can catch a transition in two successive windows, we can immediately determine the relationship between the sample bit delay and the window cycle time (See Equation 10).

Figure 23: Slow Clock Sample Register Calibration (Overlap Case)

Clock Signal	000011111111111100000000...
Clock Sample 1	$\boxed{00001111}$1111111100000000...
Clock Sample 2	00001$\boxed{11111111}$11100000000...
Clock Sample 3	0000111111$\boxed{11111100}$000000...
Clock Sample 4	000011111111111$\boxed{10000000}$0...
$\vdots$	$\vdots$
Clock Sample $m+1$	000$\boxed{01111111}$1111100000000...
Clock Sample $m+2$	00001111$\boxed{11111111}$00000000...
Clock Sample $m+3$	0000111111111$\boxed{11100000}$000...

Here again is a slow clock case where a pair of calibration clock edges never occur in a single sample window. In the two sequences shown, the pair of edges occur within a differing number of windows. The combined information from these two series of samples give us enough information to solve for the sample bit delay and window cycle time using Equation 9.

Figure 24: Slow Clock Sample Register Calibration (Differing Window Alignment)

impedance values.

From this basic algorithm, we can compute the bandwidth requirements for impedance tuning and get a basic estimate for tuning time:

- Step #2 requires loading in an impedance values. With n_i impedance transistors on each supply network, this requires $2n_i$ bits per controlled impedance pad.
- Step #3 can occur in parallel for all pads being tuned and can be done in tens of clock cycles per iteration.
- Step #4 offloads the impedance values and requires n_{sbit} bits per controlled impedance pad.

If we scan through all 2^{ni} settings for an impedance network, and have to iterate the process four times to achieve reasonable convergence, the total number of bits transfered is:

$$n_{bit} = 4 \cdot 2^{ni} \cdot n_{pads} \cdot (n_{sbit} + 2n_i) \qquad (12)$$

If we move data to and from the chip under serial TAP control, we get to move one bit per clock cycle. There will be some additional overhead for the TAP protocol, but these are small compared to the clock cycles required to move data on and off the chip. The tuning time can thus be approximated as:

$$T_{tune} = \left(4 \cdot 2^{ni} \cdot n_{pads} \cdot (n_{sbit} + 2n_i)\right) \cdot t_{tclk} \qquad (13)$$

To make this concrete, we can consider the test pad from Section 9. This pad had $n_i = 6$ and $n_{sbit} = 16$. If we further consider a component with 160 impedance controlled pins and a scan-based TAP with a 20 MHz scan clock (TCLK):

$$\begin{aligned} T_{tune} &= \left(4 \cdot 2^6 \cdot 160 \cdot (16 + 2 \cdot 6)\right) \cdot 50 \text{ ns} \\ &\approx 57 \text{ ms} \end{aligned}$$

This time can be reduced by clever arrangement of the scan operations. For instance, if we know we are going to be tuning all 160 pins in parallel, a parallel load of all 160 pins from the same $2n_i$ impedance bits would make the cost of uploading test impedance values almost negligible. For the example above, such a change would reduce the time to roughly 33 ms. In the analysis above, we assumed that it was necessary to reload both the pull-up and pull-down impedance during step #2, while only one of these impedances generally varies during a scan iteration. Simply allowing the pull-up and pull-down networks to be loaded independently would allow us to tune the impedance in 45 ms.

Of course, if a single pad's impedance and sample registers can be accessed independently, the offload time for a single pad during tuning would be:

$$\begin{aligned} T_{tune_{one\_pad}} &= \left(4 \cdot 2^6 \cdot 1 \cdot (16 + 2 \cdot 6)\right) \cdot 50 \text{ ns} \\ &\approx 360 \ \mu\text{s} \end{aligned}$$

When tuning a single pad, the scan overhead will not be as negligible. A more accurate estimate for single pad tuning time is on the order of 500 μs using a scan-based TAP.