

Introduction

An overview of digital I/O signal processing methods

I/O Performance Limitations

Input/output (I/O) has always played a crucial role in computer and industrial applications. But as signal processing became more sophisticated, problems arose that prevented reliable I/O communication. In early parallel I/O buses, interface alignment problems prevented effective communication with outside devices. And as higher speeds became prevalent in digital design, managing signal delays became problematic.

Digital Design Solutions for I/O

Digital designers turned to a host of methods to increase signal speed and eliminate I/O problems. For example, differential signal processing was employed to increase speed in chip-to-chip communications. And design methods such as signal-, source-, and self-synchronization refined inter-IC (integrated circuit) communication to provide reliable I/O at speeds demanded by the computer industry.

Introducing Multi-Gigabit Serial

Figure 1-1 shows a typical digital signal.

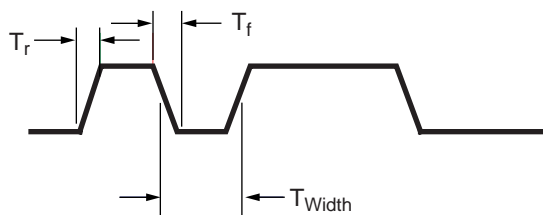


FIGURE 1-1: Standard Digital Signal

Notice the values of the time measurements listed on the diagram:

$$T_R = 20 \text{ ps}$$

$$T_F = 20 \text{ ps}$$

$$T_{\text{WIDTH}} = 0.10 \text{ ns}$$

These values represent a *very* fast waveform. Figure 1-2 adds historical signals for reference to show just how fast this waveform is.

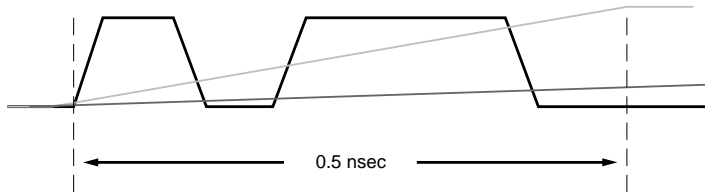


FIGURE 1-2: Adding Historical Signals

Most signals cannot even get through their rise times in five bit times of the signal. So why discuss such a signal? Because it represents the hottest trend in digital I/O—multi-gigabit serial.

This type of signal is exploding onto the market. It's finding applications in everything from local area network (LAN) equipment, to cutting edge medical imaging equipment, to advanced fighter jet technology. Multi-gigabit signals are quickly becoming key to the expanding information age. To learn about this fast-moving technological advancement, let's review the history of I/O design.

History of Digital Electronic Communication

Transistor-transistor logic (TTL) was once the premier design method. Discrete gate ICs would communicate with each other to form larger circuits that were then integrated into complex ICs such as multi-bit registers and counters. Parallel communication dominated printed circuit board (PCB) assemblies for many years. But alignment problems were too difficult for outside communications. As a result, the serial port ruled box-to-box communications as evidenced by the serial printer ports in early computers.

A critical part of learning about a new technology is learning the vocabulary. Throughout the book you will find definitions of **key terms** set aside like this.

Eventually the alignment problems were solved. High-speed parallel printer ports proliferated as parallel technologies evolved. These included Industry-Standard Architecture (ISA), Extended Industry-Standard Architecture (EISA) and Small Computer Systems Interface (SCSI), Peripheral Component Interconnect (PCI), and the smaller Personal Computer Memory Card Industry Association (PCMCIA).

Serial technology continued to coexist with parallel. Ethernet and Token Ring gained dominance in many applications. Eventually, Token Ring replaced Ethernet when it was made to work on category 5 (Cat 5) wire.

Parallel technologies struggled to accommodate new interface demands. Standards like PCI 33 evolved into PCI 66 as more exotic signaling was required. For a while, low swing standards such as high-speed transistor logic (HSTL) attempted to support parallel technology. Meanwhile, Ethernet went from 10 Mb to 100 Mb to 1000 Mb per second. Such speeds made Ethernet highly desirable for the desktop.

About this time the fractional phase detector was introduced. This technology boosted serial interface speed to the multi-gigabit range. Serial was proving to be fast and strong and it found application as a backplane technology. As serial pin count and simultaneous switching outputs (SSO) improved, multi-gigabit serial gained prominence in PCB assemblies and replaced parallel.

Basic I/O Concepts

Single-ended I/O has been the standard for years. In single-ended systems, one signal connection is made between the two ICs. This signal is compared to a specified voltage range (TTL CMOS [complementary metal oxide semiconductor]), or a reference voltage (HSTL). Sample specifications of these methods are shown in Figure 1-3.

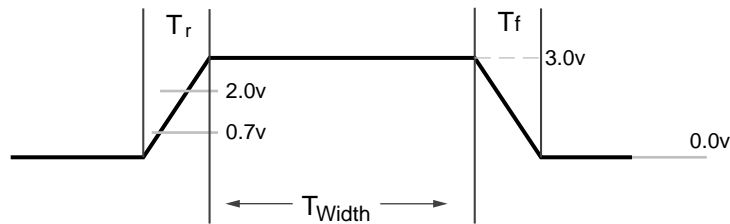


FIGURE 1-3: TTL Waveform

TABLE 1-1: LVCMOS (Low-voltage CMOS) Voltage Specifications

Parameter	Minimum	Typical	Maximum
V_{CCO}	2.3	2.5	2.7
V_{REF}	-	-	-
V_{TT}	-	-	-
V_{IH}	1.7	-	3.6
V_{IL}	-0.5	-	0.7
V_{OH}	1.9	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	-12	-	-
I_{OL} at V_{OL} (mA)	12	-	-

TABLE 1-2: HSTL Voltage Specifications

Parameter	Minimum	Typical	Maximum
V_{CC0}	1.4	1.5	1.6
V_{REF}	0.68	0.75	0.90
V_{TT}	-	$V_{CC0} \times 0.5$	-
V_{IH}	$V_{REF} + 0.1$	-	-
V_{IL}	-	-	$V_{REF} - 0.1$
V_{OH}	$V_{CC0} - 0.4$	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	-8	-	-
I_{OL} at V_{OL} (mA)	8	-	-

HSTL Class III

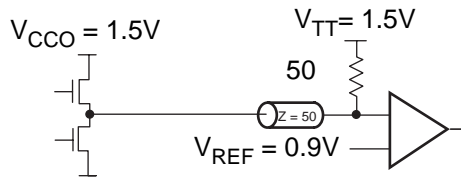


FIGURE 1-4: HSTL Voltage Diagram

Differential Signal

About the time HSTL and other low voltage swings became popular, a differential signal method began to appear on chip-to-chip communications. Differential signals had long been available, but they had been used for long transmissions, not for chip-to-chip communication on PCBs (Figure 1-5).

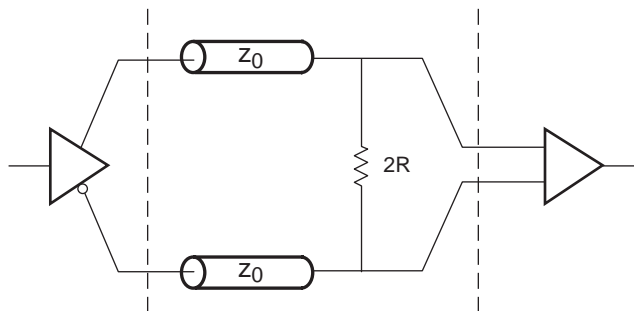


FIGURE 1-5: Differential Signal Method

As IC communication speeds increased, system and IC designers began to look for signaling methods that could handle higher speed (Figure 1-6). Differential signaling was such a method. It has several advantages over single-ended signaling. For example, it is much less susceptible to noise. It helps to maintain a constant current flow into the driving IC. And rather than comparing a voltage to a set value or reference voltage, it compares two signals to each other. Thus, if the signal referenced as the positive node has a higher voltage than the one referenced negative, the signal is high, or one. If the negative referenced signal is more positive, the signal is low, or zero. The positive and negative pins are driven with exact complementary signals as shown below.

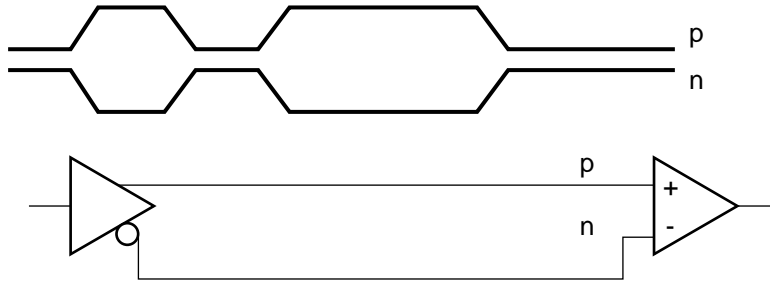


FIGURE 1-6: Signaling Methods

System-Synchronous, Source-Synchronous, and Self-Synchronous

There are three basic timing models used for communication between two ICs — system-synchronous, source-synchronous, and self-synchronous.

System-Synchronous

This method as shown in Figure 1-7 was the most common for many years. It seems very simple until we look at the timing model in Figure 1-8. The shaded boxes represent delays that must be accounted for and balanced to ensure a reliable receiving circuit.

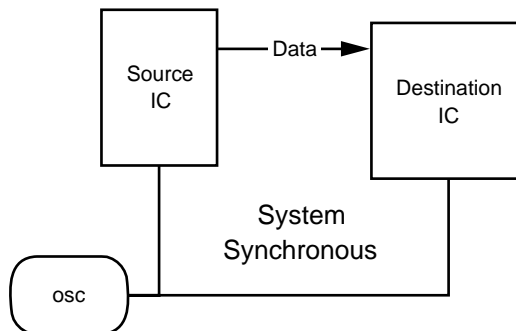


FIGURE 1-7: System-Synchronous Diagram

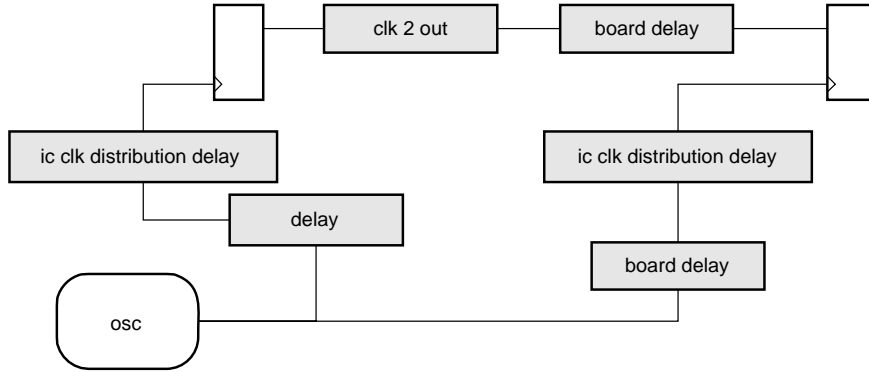


FIGURE 1-8: System-Synchronous Timing Model

System-Synchronous: Communication between two ICs where a common clock is applied to both ICs and is used for data transmission and reception.

Source-Synchronous

For years most signal delays were ignored because they were so small compared to the available time. But as speeds increased, managing delays became more difficult, then impossible. One way to improve the problem was to send a copy of the clock along with the data. This method is called source-synchronous (Figure 1-9) and it greatly simplified the timing parameters.

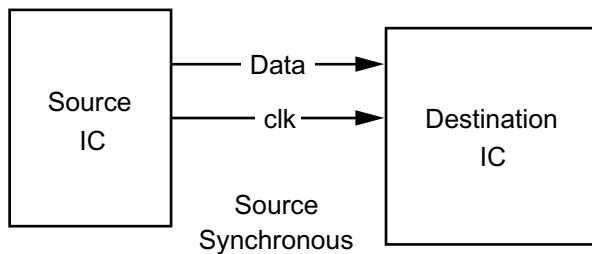


FIGURE 1-9: Source-Synchronous Diagram

The output time of the forwarded clock is adjusted so that the clock transitions in the middle of the data cell. Then the trace lengths of the data and clock lines must be matched. But there are some draw-

backs. The received data on the destination IC must be moved from the received clock domain to a global IC clock.

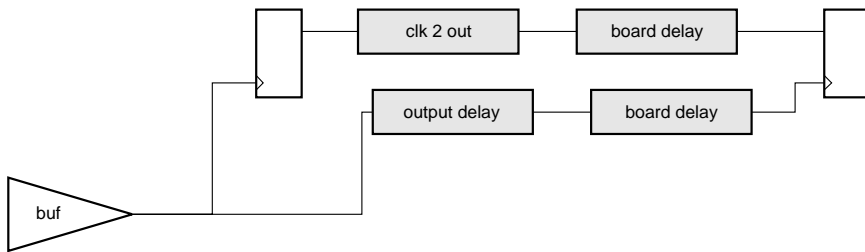


FIGURE 1-10: Source-Synchronous Timing Model

Source-Synchronous: Communication between two ICs where the transmitting IC generates a clock that accompanies the data. The receiving IC uses this forwarded clock for data reception.

Clock Forwarded: Another term for source-synchronous.

Source-synchronous design results in a marked increase in the number of clock domains. This introduces timing constraint and analysis complications for devices such as a Field Programmable Gate Array (FPGA) with limited clock buffers, and an Application-Specific Integrated Circuit (ASIC) where each clock tree must be custom designed. The problem is aggravated on large parallel buses where board design limitations often force the use of more than one forwarded clock per data bus. Hence, a 32-bit bus may require four, or even eight forwarded clocks.

Self-Synchronous

The self-synchronous model is shown in Figure 1-11. Here, the data stream contains both the data and the clock.

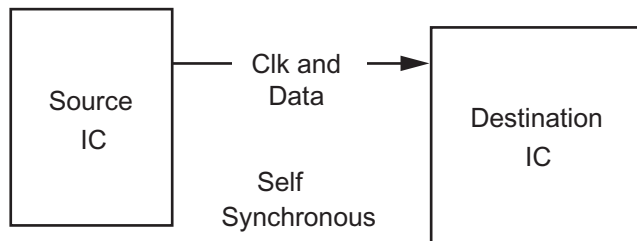


FIGURE 1-11: Self-Synchronous Diagram

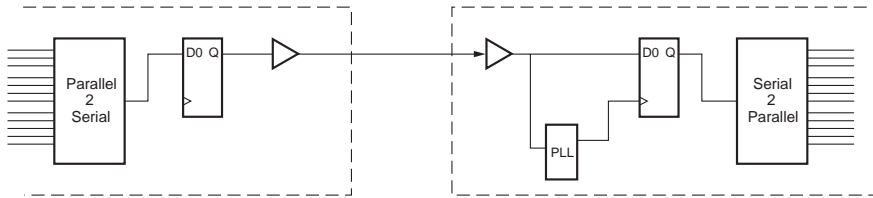


FIGURE 1-12: Self-Synchronous Timing Model

Self-Synchronous: Communication between two ICs where the transmitting IC generates a stream that contains both the data and the clock.

The three main blocks of a self-synchronous interface are parallel-to-serial conversion, serial-to-parallel conversion, and clock data recovery.

Parallel-to-Serial Conversion

There are two main methods of parallel-to-serial conversion—a loadable shift register and revolving selectors. Simple logic representations of these methods are shown in Figure 1-13.

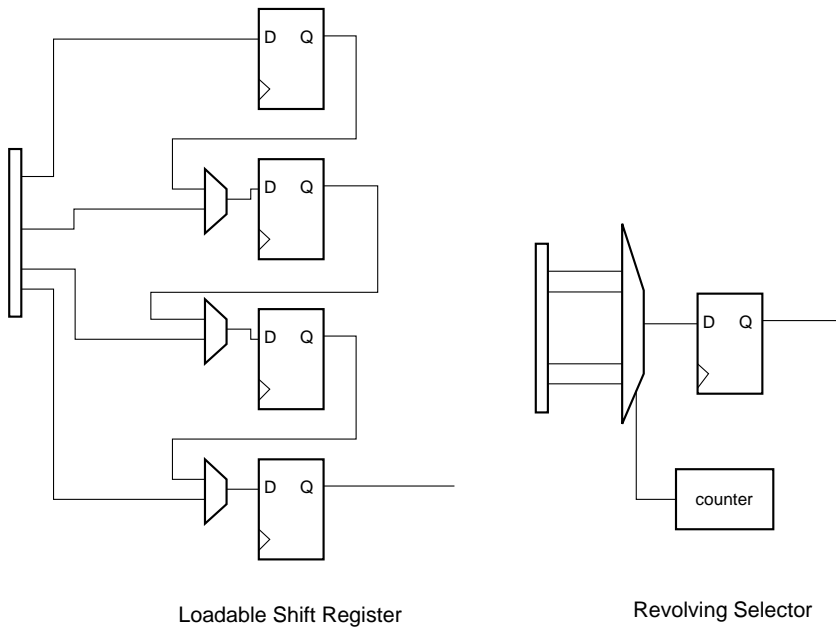


FIGURE 1-13: Parallel-to-Serial Conversion Processes

Serial-to-Parallel Conversion

The serial-to-parallel process is just the opposite as shown in Figure 1-14.

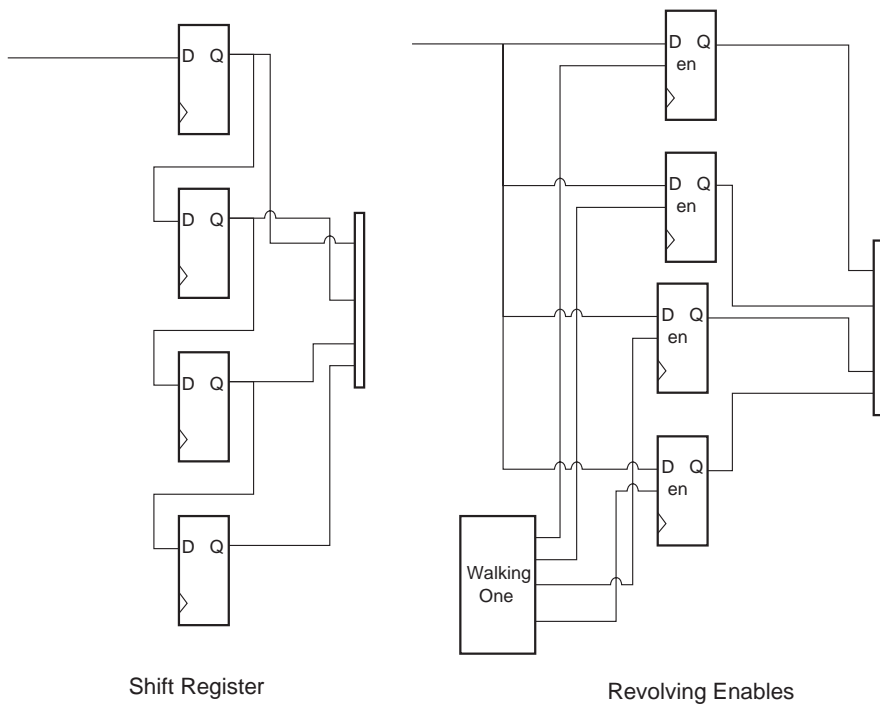


FIGURE 1-14: Serial-to-Parallel Conversion Processes

Clock/Data Recovery

The clock recovery process (Figure 1-15) does not provide a common clock or send the clock with the data. Instead, a phased locked loop (PLL) is used to synthesize a clock that matches the frequency of the clock that generates the incoming serial data stream.

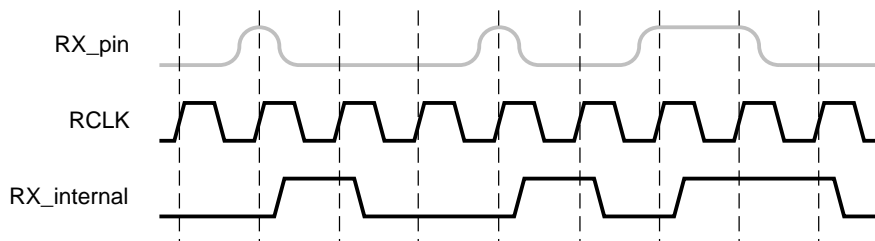


FIGURE 1-15: Clock/Data Recovery Waveform

PLL: A phased locked loop is a circuit that takes a reference clock and an incoming signal and creates a new clock that is locked to the incoming signal.

Parallel Transfers

In parallel transfers, additional control lines are often used to give different meanings to the data. Examples include data enables and multiplexing both data and control data onto the same bus.

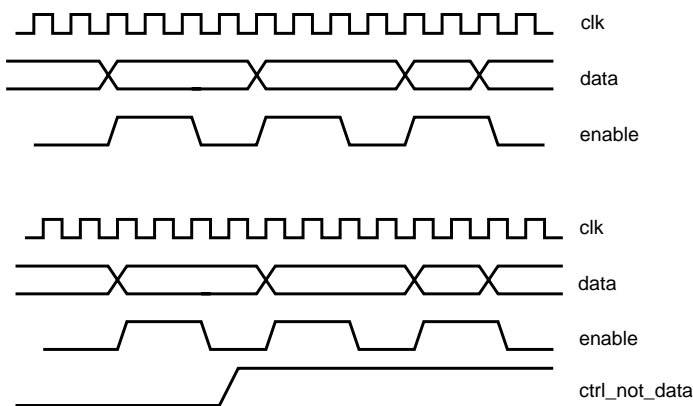


FIGURE 1-16: Parallel Transfer Example

In the serial domain, flags or markers are created to set data apart from non-data that is normally referred to as idle. Flags can also be used to mark different types of information such as data and control.

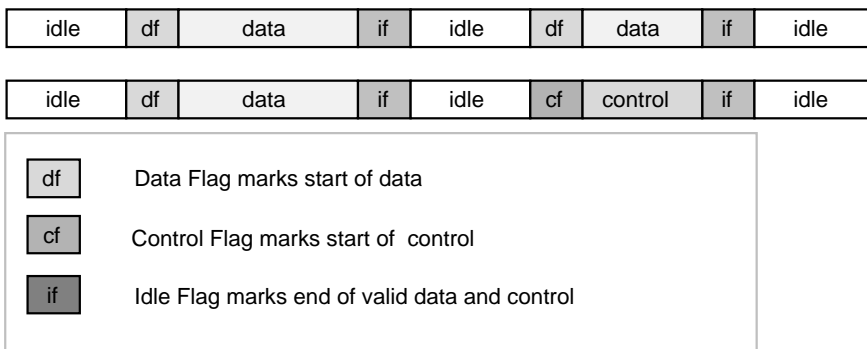


FIGURE 1-17: Serial Domain Transfers Example

Constant I/O Improvement

Industrial requirements for bandwidth and speed have demanded constant improvements in I/O design. As parallel and serial I/O have fought for prominence in chip and device communication, both have benefited from design methods that yielded vastly increased speeds. The use of digital design methods such as differential and synchronous signal processing and parallel transfers have ensured continued improvement in I/O performance for home and industry.