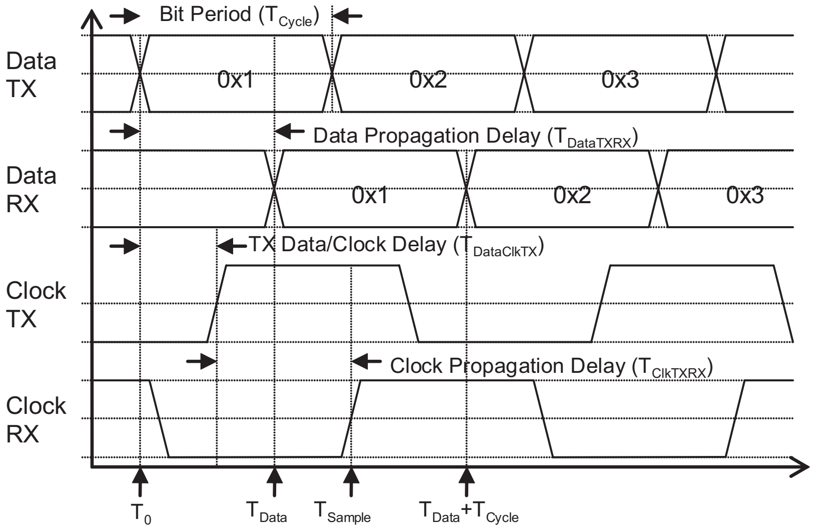
Fundamentals of Digital Communications Systems

Marcus Müller



During the last ten years, most major communications and broadcast systems and many other systems were converted from analog to digital. Examples of digital systems that we use every day include mobile phones, television, radio, and of course the Internet. CDs and MP3s are replacing records and tapes, and the number of digital cameras sold this year exceeded the number of analog cameras by a factor of three. In this chapter, you will see some of the basic building blocks that make all of these digital systems work.

The material in this chapter is intended to provide a background that will be useful when studying digital communications test and measurement techniques described in later chapters.

We start with a discussion of a basic digital communications link, cover the most commonly used clocking architectures, discuss line-coding methods, and conclude with special techniques for high-speed serial transmission systems.

1.1 Introduction

The most important aspect of any digital communications system is the required transmission speed. Just how much data needs to be transmitted, and how fast? The variability is huge, even within a single system: The keyboard interface of a typical PC, for example, runs at several kilobits per second, which is still significantly faster than anyone can type. However, the fastest interface available for graphics adapters is not nearly fast enough for the newest games, even at 40 Gbit/s (which is the accumulated bandwidth of a PCIe x16 link, the current standard for graphics adapters).

The second, equally important aspect is the link distance. How far apart are sender and receiver? Again, there is huge variability: The main processor of a computer communicates with its main memory over a distance that's usually less than 10 cm. But when you type a URL into a Web browser, you communicate with a server that's potentially on a different continent.

Generally, digital transmission becomes harder when the transmission speed and link distance increase. A measure for the effort required to make a digital communications link work is the bandwidth-distance product. An old telegraph, for example, transmitted about 100 bit/s, over a maximum distance of 20 km. The radio downlink from the *Voyager* spacecraft transmits data slightly faster, at 160 bit/s, but over an incredible distance of 14.821 billion km. The much larger bandwidth-distance product of the spacecraft link can be achieved only with incredible effort.

Every digital link consists of three components: a sender, a transport medium, and a receiver. Usually, the medium is defined first, depending on the required link bandwidth, the distance between transmitter and receiver, and economic considerations. Electrical links are still the most common type; they come in a great variety, ranging from bond wires within an integrated circuit package to printed circuit board traces on a motherboard to Ethernet cables connecting office computers. Fiber-optic cables are used for very high bandwidth connections in network and storage environments, but it seems as if "fiber to the home" might be replaced by wireless links in the near future.

1.2 System Architectures

1.2.1 Synchronous Systems

The basic synchronous digital transmission system uses a central clock that is distributed to both the transmitter (TX) and the receiver (RX) (Figure 1–1). On every clock edge, the transmitter latches the incoming data, which then travels down the transmission line toward the receiver. The receiver samples the data on the next clock edge. Short-distance synchronous systems, for example between a processor and its memory, are often parallel: Multiple data lines are clocked together.

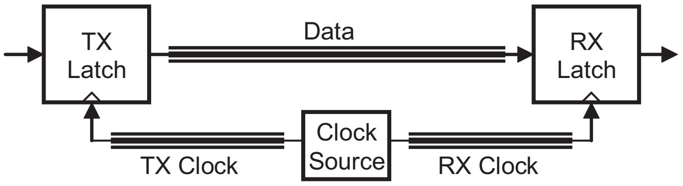


Figure 1–1 Block diagram of a synchronous system with a common clock source

Figure 1–2 shows the timing diagram for such a synchronous system, with all the relevant delays: the propagation delay of the clock signal from the clock source to the TX latch (T_{ClkTX}) and the RX latch (T_{ClkRX}), the time it takes the TX to latch the data ($T_{ClkDataOut}$), and the propagation delay of the data path ($T_{DataTXRX}$). For the sake of simplicity, we will not include timing uncertainties in our analysis. From these delays, we can calculate T_{Data}, the time when the data arrives at the receiver latch:

$$T_{Data} = T_{ClkTX} + T_{ClkDataOut} + T_{DataTXRX} \qquad \textbf{(Equation 1–1)}$$

and also T_{Sample}, the time when the receiver latch will sample the data:

$$T_{Sample} = T_{ClkRX} + T_{Cycle} \qquad \textbf{(Equation 1–2)}$$

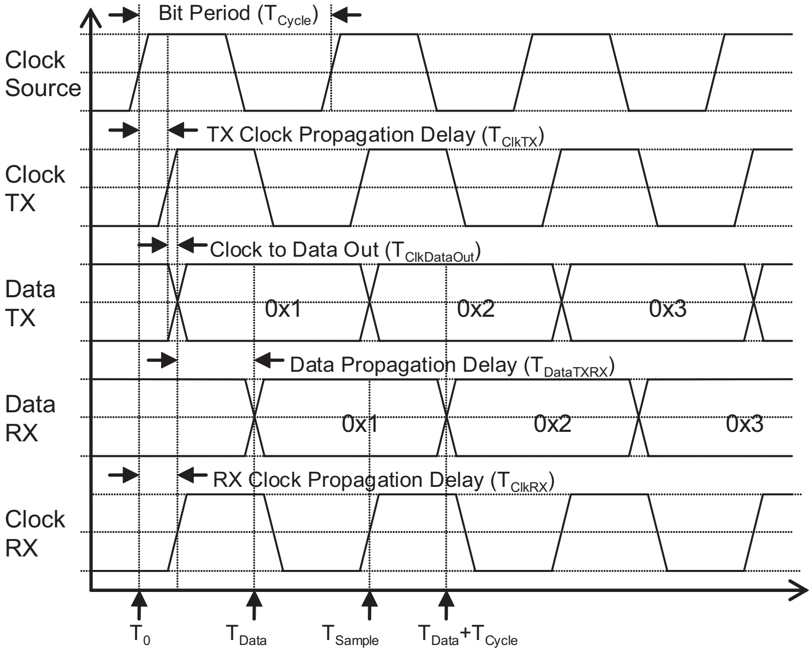

Bit Period (T_Cycle)
Clock Source
TX Clock Propagation Delay (T_ClkTX)
Clock TX
Clock to Data Out (T_ClkDataOut)
Data TX
0x1
0x2
0x3
Data Propagation Delay (T_DataTXRX)
Data RX
0x1
0x2
0x3
RX Clock Propagation Delay (T_ClkRX)
Clock RX
T_0
T_Data
T_Sample
T_Data+T_Cycle

Figure 1–2 Timing diagram for a synchronous system with a common clock

An additional requirement of the receiver latch is that the incoming data is stable for some time before and after the sampling clock edge; it requires a positive setup time (T_{Setup}) and hold time (T_{Hold}). How long the data at the receiver latch is stable before sampling is equal to the time difference between T_{Sample} and T_{Data}. In order to maintain the setup time requirement, this value has to be larger than the setup time:

$$T_{Sample} - T_{Data} > T_{Setup} \qquad \textbf{(Equation 1–3)}$$

Then the setup time margin is

$$\begin{aligned} M_{Setup} &= T_{Sample} - T_{Data} - T_{Setup} \\ &= T_{ClkRX} + T_{Cycle} - T_{ClkTX} - T_{ClkDataOut} - T_{DataTXRX} - T_{Setup} \end{aligned} \qquad \textbf{(Equation 1–4)}$$

From this, we can calculate the minimum cycle time, by setting the setup margin to zero:

$$T_{Cycle,min} = -T_{ClkRX} + T_{ClkTX} + T_{ClkDataOut} + T_{DataTXRX} + T_{Setup} \qquad \textbf{(Equation 1–5)}$$

If a system has insufficient setup margin, we can increase either the cycle time (make the system slower) or the RX clock propagation delay, or we can decrease either the TX clock propagation delay or the data propagation delay.

How long the data at the receiver latch is stable after sampling is equal to the time difference between T_{Data} plus one cycle, minus T_{Sample}. This value has to be larger than the hold time:

$$T_{Data} + T_{Cycle} - T_{Sample} > T_{Hold} \qquad \textbf{(Equation 1–6)}$$

Then the hold time margin is

$$\begin{aligned} M_{Hold} &= T_{Data} + T_{Cycle} - T_{Sample} - T_{Hold} \\ &= T_{ClkTX} + T_{ClkDataOut} + T_{DataTXRX} - T_{ClkRX} - T_{Hold} \end{aligned} \qquad \textbf{(Equation 1–7)}$$

Note that the hold time margin is independent of the cycle time; the hold time requirement doesn't relax if the system runs at a slower speed. In order to gain hold time margin, we can decrease the RX clock propagation delay, or we can increase either the TX clock propagation delay or the data propagation delay.

Let's consider an example: an 8-bit parallel synchronous system with 74ACT646 registered transceivers (Figure 1–3). The distance between the transmitter and receiver is 6 inches, which is equivalent to a propagation delay of approximately 1.0 ns on an FR4 printed circuit board. For simplicity, we assume that the clock source is exactly in the middle between transmitter and receiver, so that both T_{ClkTX} and T_{ClkRX} are 0.5 ns. The 74ACT646 has a specified worst-case setup time of 5.0 ns, a hold time of 0.0 ns, and a clock-to-data-out time of 12.0 ns. From Equation 1–5, we calculate the minimum cycle time as 18.0 ns, which gives us a maximum operating frequency of 55.55 MHz; the hold time margin for this setup is 13.0 ns. If we place the clock source at the transmitter (so that T_{ClkTX} equals 0.0 ns and T_{ClkRX} equals 1.0 ns), the mini-

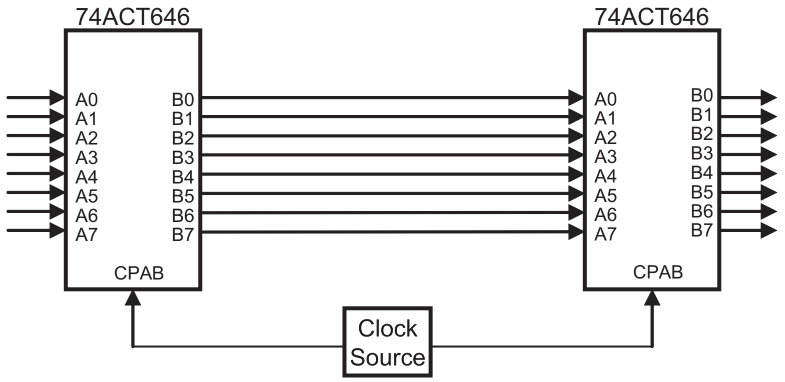


Figure 1–3 Example of a parallel synchronous system with 74ACT646 octal registered transceivers. Both devices are set to latch data from port A (A0–A7) to port B (B0–B7) on a positive edge on the AB clock pulse input (CPAB).

mum cycle time is only 17.0 ns, so we can operate at frequencies up to 58.82 MHz, still without violating the hold time requirement: The hold time margin for this configuration is 12.0 ns.

One of the most widely used parallel synchronous systems is the Peripheral Component Interconnect (PCI) bus, designed to attach peripherals to computers. PCI is a multidrop configuration, where multiple receivers are connected to the same transmitter. And in multipoint applications, bidirectional transceivers are attached to a common data bus (Figure 1–4). A bus master is responsible for maintaining bus integrity; for example, it ensures that no two systems send data at the same time. Setup and hold time requirements need to be fulfilled for all combinations of send and receive, which is a further limitation on the speed that is achievable with such a configuration. Different variants of PCI run at 33 MHz and 66 MHz, with 32 or 64 parallel data lines. PCI-X increased the signaling rate even further (to 133 MHz, 266 MHz, and even 533 MHz) but never became widely used in consumer products because of costs associated with the complicated signal routing.

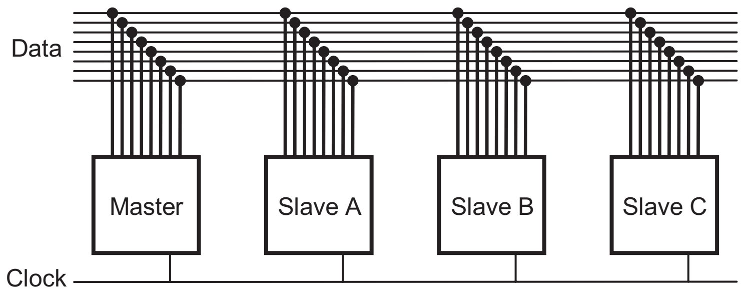


Figure 1–4 Block diagram of a bidirectional multipoint parallel bus (e.g., PCI)

1.2.2 Source Synchronous Systems

In source synchronous systems, the sampling clock is sent along with the data by the transmitter, rather than a central clock source as in synchronous systems. Figure 1–5 shows a generic block diagram. The transmitter has its own clock source, which generates edges for the TX data latch and the clock that is sent to the receiver for sampling. The delay element in the TX clock path ensures that the clock edge arrives at the receiver later than the data, which is required for correct sampling.

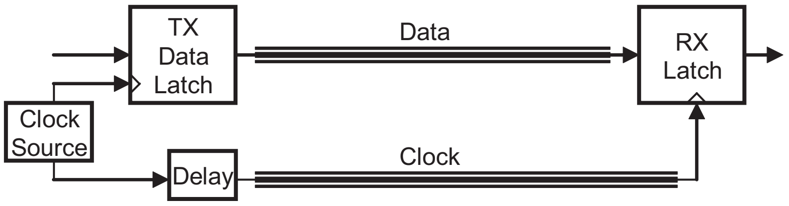


Figure 1–5 Block diagram of a source synchronous system

Source synchronous systems can operate at significantly higher speeds than synchronous systems. The reason for this becomes clear when we look at the timing diagram for the system (Figure 1–6). The relevant delays are the propagation delay of the data ($T_{DataTXRX}$) and clock ($T_{ClkTXRX}$) and the delay between data and clock at the transmitter ($T_{DataClkTX}$). The two latches at the transmitter do have clock-to-data-out times, but we simply included them in the propagation delays; we've done the same for the clock distribution within the transmitter. The time when the data becomes valid at the receiver is

$$T_{Data} = T_{DataTXRX} \qquad \textbf{(Equation 1–8)}$$

and the sample time is

$$T_{Sample} = T_{DataClkTX} + T_{ClkTXRX} \qquad \textbf{(Equation 1–9)}$$

Note that the cycle time disappeared from the two equations; source synchronous systems don't have a theoretical frequency limit. Setup and hold time requirements still need to be satisfied, however. From the setup time and hold time requirements (Equations 1–3 and 1–6), we calculate the setup time margin:

$$M_{Setup} = T_{Sample} - T_{Data} - T_{Setup} = T_{DataClkTX} + T_{ClkTXRX} - T_{DataTXRX} - T_{Setup}$$

(Equation 1–10)

and the hold time margin:

$$M_{Hold} = T_{Data} + T_{Cycle} - T_{Sample} - T_{Hold} = T_{DataTXRX} + T_{Cycle} - T_{DataClkTX} - T_{ClkTXRX} - T_{Hold}$$

(Equation 1–11)

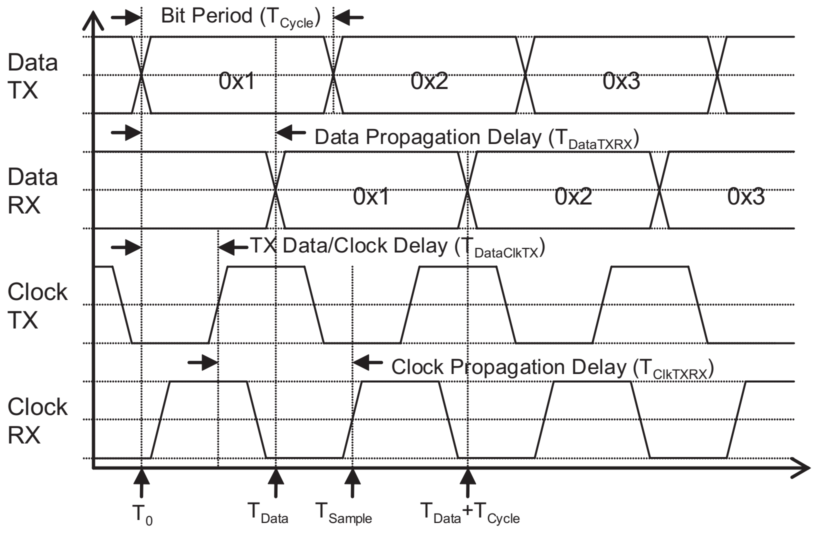


Figure 1–6 Timing diagram for a source synchronous system

Because the data-to-clock delay at the transmitter is controlled by the delay element, we can make every source synchronous system work, provided that the sum of the setup and hold times doesn't exceed the cycle time. At very high speeds, however, it becomes increasingly difficult to control the skew between the data path and the clock path, especially if data is transmitted in parallel. Practical source synchronous systems operate at data rates up to 1 GHz.

1.2.3 Source Synchronous Systems with Double Data Rate

A variant of source synchronous transmission uses a half-rate clock and latches data at both the rising and the falling edges. Because the data is transmitted at double the speed relative to a normal clock, this variant is called double data rate (DDR) signaling. Figure 1–7 shows an example of a timing diagram for such a system. The timing relationships are almost the same as before, with one difference: Both positive and negative clock edges can be used as sampling references.

Because of the reduced clock speed, signal routing is simplified, and lower bandwidth connectors can be used. This is one of the reasons why DDR is used, for example, in high-speed memory interfaces such as DDR-2 SDRAM, with clock speeds up to 400 MHz and corresponding data transfer rates up to 800 Mbit/s.

1.2.4 Forwarded Clock Systems

Forwarded clock systems are very similar to source synchronous systems (Figure 1–8). The main idea of the forwarded clock is that the clock path from the transmitter to the receiver experiences the exact same noise and jitter as the data path.

The first major difference of the forwarded clock architecture compared to the source synchronous architecture is that the delay element in the clock path resides in the receiver rather

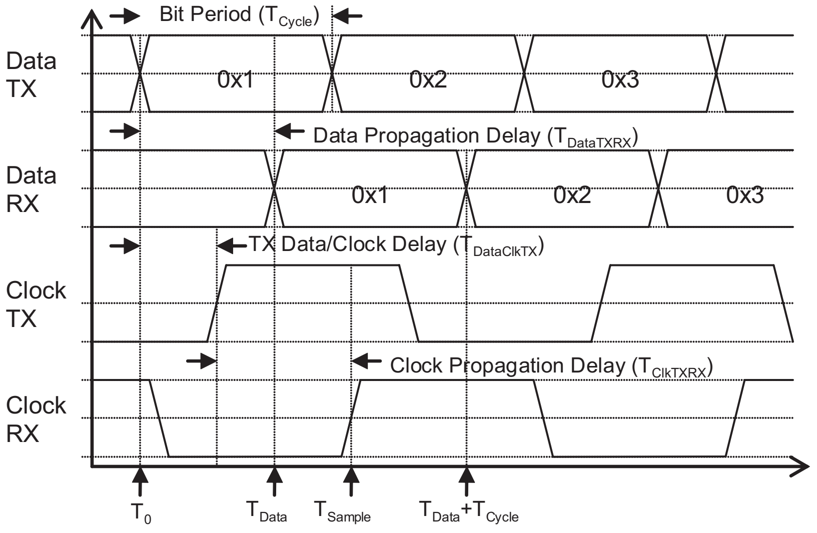


Figure 1–7 Timing diagram for a source synchronous system with a double data rate clock

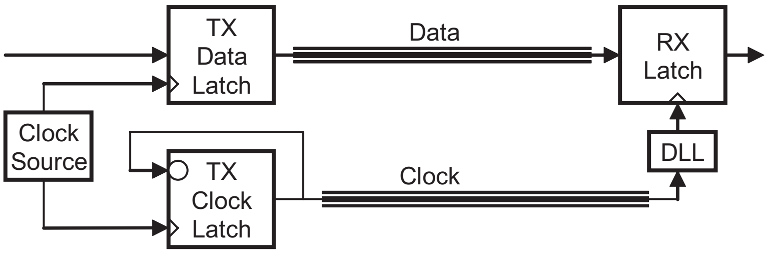


Figure 1–8 Block diagram of a forwarded clock system

than in the transmitter. This is to make sure that any jitter that occurs during the transmission impacts both data and clock and hence cancels out. The delay element on the receiver is a delay locked loop (DLL) in most cases, which increases the flexibility of the system because it automatically adjusts the delay between the data path and the clock path.

The second major difference is that there are now two latches on the transmitter side: one for the data and one for the clock. The exact same type of latch and driver are used for the clock and the data. This is to ensure that any negative effects that the driver may have on the signal (e.g., thermal drift) affect both the data path and the clock path and cancel out.

1.2.5 Embedded Clock Serial Systems

Embedded clock systems transmit only the serial data stream, and the receiver extracts the sampling clock automatically from the data (Figure 1–9). The main advantage of this architecture is

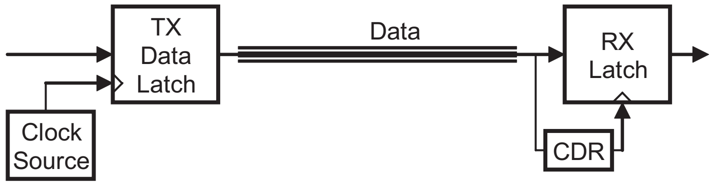


Figure 1–9 Block diagram of an embedded clock system

that propagation delays and skew are nonissues. The clock data recovery (CDR) circuit at the receiver takes care of the correct phase alignment between data and clock. This enables serial data signaling at very high rates, up to 10 Gbit/s and beyond. Also, the CDR circuit can track some variations in clock speed and other low-frequency and time variations, which makes embedded clock systems very robust.

Because the link between the transmitter and the receiver consists of only one transmission line, the possible routing density is greatly increased over parallel source synchronous systems. Serial data can be easily transmitted over thin and flexible cables. Long-distance optical communications systems use this clocking scheme almost exclusively.

However, the price to pay for the flexibility and the high data rates is increased complexity of the transmitter and especially the receiver. The main building blocks of a serial embedded clock system are the parallel-to-serial conversion at the transmitter and receiver, the reference clock generation, and the clock data recovery. Integrated circuit designs are commonly available, though, so designs of this type can be inexpensive and straightforward. We will look at these building blocks in more detail in the following subsections.

1.2.5.1 Serializer and Deserializer

In most cases, the data within both the transmitter and the receiver is kept parallel and converted to and from serial format only for the data transmission over the serial link. The components that perform this conversion from parallel to serial and back are called serializer and deserializer (SERDES) components. SERDES components often integrate the TX and RX latches shown in Figure 1–9.

In Figure 1–10, we show an example implementation of a 4:1 parallel-to-serial converter. The heart of the serializer is a shift register, consisting of the latches L0 to L3. The shift register is clocked by the serial clock, at the serial data rate. The inputs into the latches are multiplexed, and the control input for the multiplexers selects either the shift register chain or the parallel data bits D0 to D3 from the parallel input latch. The clock for the parallel latch is the serial clock divided by four. The control signal needs to select the parallel input for one cycle of the serial clock, and the shift register chain for the next three cycles.

Figure 1–11 shows the corresponding deserializer. The serial data is clocked through the shift register (L0 to L3) and latched into the parallel output latch (D0 to D3) every four cycles of the serial clock.

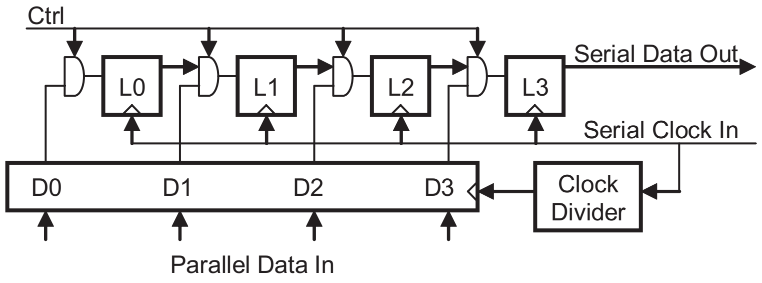


Figure 1–10 Implementation example for a 4:1 shift register serializer

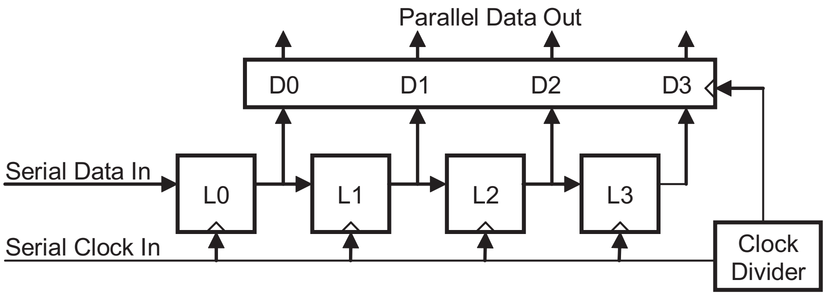


Figure 1–11 Implementation example for a 1:4 shift register deserializer

The drawback of this rather simplistic SERDES design is that the phase of the incoming data is not known. If the serializer and deserializer operate back to back, the parallel data is not guaranteed to be recovered with the same phase. Figure 1–12 shows an example of this behavior: The parallel data at the output is rotated by one bit relative to the parallel input data. More advanced SERDES architectures provide word synchronization features that ensure that the parallel data phase is correct.

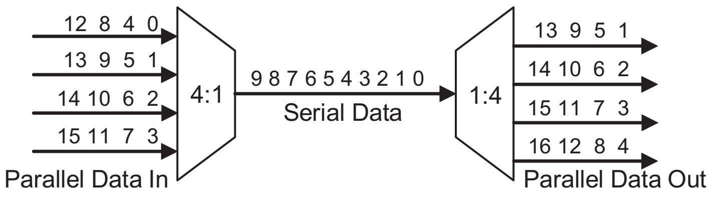


Figure 1–12 Serializer and deserializer (represented by the trapezoids) in back-to-back mode. Deserializer is out of phase.

1.2.5.2 Reference Clock Generation

Both the serializer and deserializer require an at-speed clock signal. At the receiver, this clock is supplied by the clock data recovery circuit, but to avoid the routing of a high-speed clock signal across the system, the transmitter needs to create its own high-speed reference. It is usually generated from a lower-speed system reference clock with the help of a multiplying phase locked loop (PLL) circuit.

1.2.5.3 Clock Data Recovery

The CDR circuit extracts the sampling clock from the serial data stream, adjusting both the phase and frequency in the process to ensure proper sampling. There are two types of CDR circuits: analog and digital.

Analog CDR circuits (Figure 1–13) consist of a phase detector, a loop filter, and a voltage-controlled oscillator (VCO). The phase detector compares the phase of the serial data with the phase of the VCO output. The phase detector output is then low-pass filtered and passed on to the control input of the VCO and therefore tracks the incoming data. The dynamic properties of an analog CDR circuit depend on all three components, but the loop filter certainly has the largest impact; its characteristics determine how fast the CDR circuit locks on the data at start-up (lock time), how much frequency and phase variation can be tracked (tracking range), and how quickly the CDR circuit responds to frequency and phase changes at the input (loop bandwidth). The quality of the output clock depends mainly on the properties of the VCO and its support circuitry (e.g., the power supply): The lower the phase noise of the VCO, the cleaner the clock.

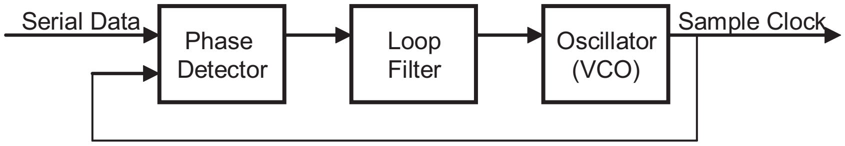


Figure 1–13 Block diagram of an analog CDR circuit

Digital CDR circuits (Figure 1–14) don't have their own oscillator and therefore require a reference clock. The high-speed sample clock is generated from the lower-speed reference clock with a multiplying PLL, using the exact same circuit used in the transmitter. The phase interpolator adjusts the relative phase between the serial data and the clock such that the deserializer can properly sample the data. Because there is no VCO needed, digital CDR circuits are relatively cheap and therefore preferred in many applications. Once the distance between transmitter

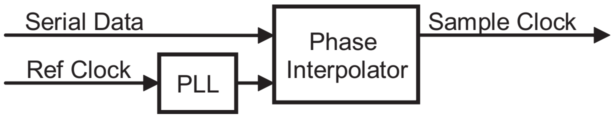


Figure 1–14 Block diagram of a digital CDR circuit

and receiver is too long, however, the effort for the distribution of the reference clock exceeds the cost of the VCO.

1.2.5.4 Special Topics in Embedded Clock Systems

Both analog and digital clock data recoveries require a minimum number of transitions in the incoming data stream, or they will lose frequency and phase lock. Since random binary data can contain long streams of consecutive one or zero bits, the data has to be altered to guarantee the minimum transition density. But too many transitions can be problematic, too, because the loop filter characteristic and thus the CDR bandwidth can change. For this reason, data for embedded clock transmission usually is encoded. We will discuss the most important coding schemes in Section 1.3.

Since both the transmitter and receiver in an embedded clock system create their own high-speed clocks, the two ends of the transmission system will often run with a slight frequency offset. If the transmitter is faster than the receiver, data can get lost; if the transmitter is slower, it runs out of data to pass to the parallel side. There are several methods to introduce elasticity to compensate for these frequency offsets. One option is to use first-in-first-out type buffers on both ends; however, that's possible only if both TX and RX operate on the exact same average frequency, for example, because they derive their high-speed clocks from the same reference. An alternative is to add comma characters to the data; these are special short bit sequences that do not carry any payload, and can therefore be discarded or inserted as required by the frequency offset.

1.2.6 Spread Spectrum Clocking

Most digital transmission systems operate at frequencies that are regulated, for example, because they are used for TV and radio broadcasting, mobile phone systems, or other radio frequency applications. In order to limit interference, agencies such as the Federal Communications Commission in the United States have put strict limits on the energy that a device may emit.

Shielding a digital communications system is often not practical, either because it is too difficult mechanically or because of cost considerations. Many systems therefore use a spread spectrum PLL, which adds a small amount of low-frequency modulation to the central clock source. The modulation reduces the peak emissions by spreading the emitted energy over a wider frequency band; however, it does not reduce the total emissions of the system.

If the modulation parameters are chosen carefully such that all parts of the system can track the spread spectrum clock, the system performance is not affected; typical values are 0.5% and 30 kHz. Many systems (e.g., PCI express) use an asymmetric approach: The frequency is modulated only downward, in order to keep the maximum frequency below the design limit.

1.3 Line Coding of Digital Signals

When binary data is sent through a link, it is represented by a physical quantity in the transport medium. In electrical links, that's usually a voltage or current; optical systems use the intensity of light; and wireless radio links often use the phase and frequency of a signal carrier. Line coding determines how the binary data is represented on the link.

Numerous coding schemes are available, and which one is best for any given application depends on many factors. Coding can influence the frequency spectrum, the direct current content, and the transition density of the resulting data stream. Coding efficiency determines the required link bandwidth, and the cost of implementation depends on the complexity of the code.

1.3.1 Properties of Binary Data

1.3.1.1 Mark Density

The mark density (MD) of a binary data pattern is defined as the number of one bits in the pattern, divided by the length of the pattern:

$$MD = \frac{N_{One}}{N_{One} + N_{Zero}}$$ **(Equation 1–12)**

where N_{One} is the number of ones in the pattern, and N_{Zero} is the number of zeros. The mark density ranges from 0.0 to 1.0, where the extremes are marked by all-zeros (N_{One} equals 0) and all-ones data (N_{Zero} equals 0). Random data is exactly at the middle of the range: It contains as many one bits as zero bits, and its long-term mark density is therefore 0.5. If we look only at a subsection of the random data pattern, however, its mark density can be very different.

If we represent a zero bit by 0.0 and a one bit by 1.0, the mark density is equal to the time average over the pattern. It is therefore a direct measure for the DC content of the signal. A pattern with a mark density of 0.5 is therefore also called a DC-balanced pattern. DC balance is an important property in some applications; if it is required to maintain a DC level in the link, then amplifiers and other system components need to be DC coupled, often leading to a more complicated and problematic design.

1.3.1.2 Transition Density

The transition density (TD) of a data pattern is defined as the number of transitions in the pattern, divided by the length of the pattern:

$$TD = \frac{N_T}{N_{One} + N_{Zero}}$$ **(Equation 1–13)**

where N_T is the number of transitions in the pattern, N_{One} is the number of ones, and N_{Zero} is the number of zeros. The transition density ranges from 0.0 to 1.0, where the extremes are marked by static patterns (all-zeros or all-ones) and toggle patterns. Random data is again exactly at the middle of the range: Because the probability that two consecutive bits are identical is 0.5, the transition density is 0.5, too.

1.3.1.3 Run Length Distribution

The run length distribution of a data pattern gives the relative probabilities for runs of identical consecutive bits. Longer runs create stress in many applications, because of either excessive intersymbol interference (ISI) or baseline wander due to local disparity.

1.3.2 Binary Line Codes

1.3.2.1 Non-Return-to-Zero Code

The non-return-to-zero (NRZ) format is the prototypical representation of binary data: A logical zero state is transmitted as one signal level, and a logical one state as another level. Levels change at bit boundaries only if the bit value changes and remain stable for the entire duration of the bit period. If the level representing the zero logical bit state is lower than the level for the one state, we call this positive logic, and the respective levels are then called low level and high level. NRZ coding is essentially free because binary data is already stored in this format in CPUs and other digital devices. It is therefore the most commonly used coding scheme and the reference for all other coding schemes in terms of signal properties, efficiency, and implementation effort.

NRZ signals always have a clock signal associated with them, even if it is not transmitted along with the data. Figure 1–15 shows the NRZ representation of a short data sequence, together with a clock signal. Note how the data signal changes on the falling edge of the clock; the receiver samples it on the rising edge. There are also systems that work with an inverted clock. The data then changes on the rising edge, and the receiver samples at the falling clock edge. The clock signal for NRZ transmission usually runs at the base frequency of the data: for a 10 Gbit/s signal, the clock rate is 10 GHz (single data rate, SDR). A variant of NRZ transmission uses a clock signal at half rate (5 GHz for 10 Gbit/s), and the receiver samples the data both at the rising and falling edges of the clock. This is called double data rate (DDR) transmission.

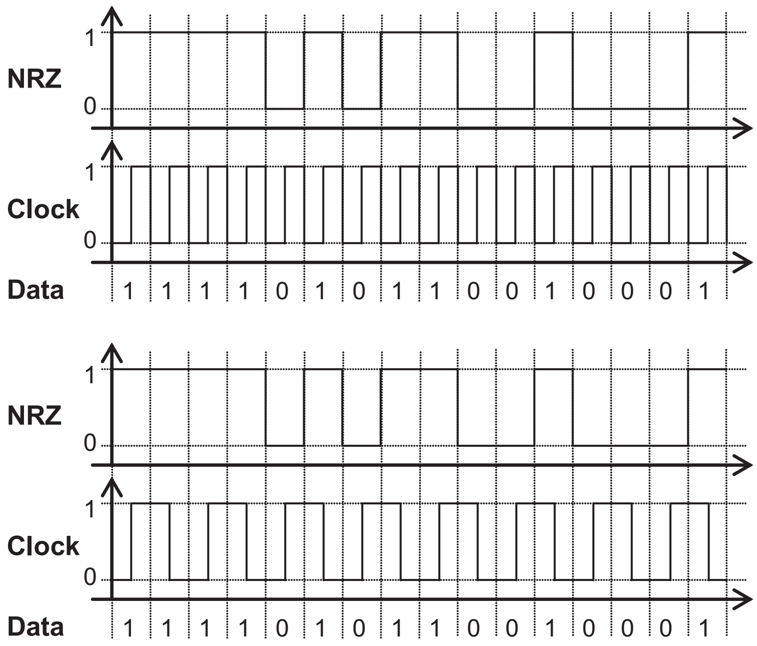


Figure 1–15 NRZ coding of a short data sequence (PRBS 2^4-1). Top: single data rate clock. Bottom: double data rate clock.

The properties of NRZ-formatted data depend entirely on the data itself. The drawback of NRZ coding is that the DC content, frequency spectrum, and transition density depend on the data sequence. Long runs of zeros or ones cause problems in some applications because of effects such as baseline wander and ISI or because there are not enough transitions for clock data recovery.

Figure 1–16 shows the power spectral densities of two short NRZ-formatted data sequences. Note how both spectra have zero power at multiples of the signal base rate (e.g., 1 GHz, 2 GHz, 3 GHz). The PRBS spectrum follows the typical sinc envelope, with nulls at multiples of the data rate. Because of the very fast rise times that we used to create the spectrum, there is significant spectral content at very high frequencies. The spectrum for the toggle pattern equals that of a 500 MHz square wave. The spectra of all-zeros or all-ones patterns are zero, with the exception of a DC value.

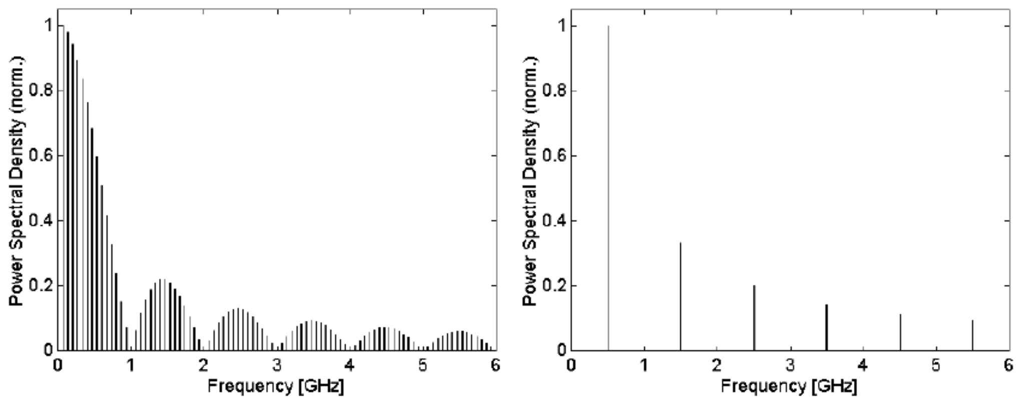


Figure 1–16 Power spectral density for NRZ-formatted data at 1 Gbit/s. Left: PRBS 2^4-1. Right: Toggle pattern (101010 . . .). Power density is normalized to a maximum power of 1.0.

1.3.2.2 Return-to-Zero Code

The return-to-zero (RZ) code represents the zero logical state as a static low level and the one state as a short high-level pulse. The signal always returns to the level representing a zero state immediately after the high level, hence the name. RZ signals can be easily created from NRZ signals, by a binary AND of the NRZ and a clock. The width of the pulses depends on the duty cycle of the clock. Figure 1–17 shows the RZ representation of a short data sequence, with 50% and 25% duty cycles.

RZ coding is used primarily in optical transmission systems because it minimizes power consumption and the effects of system dispersion on optical signal distortion. Consecutive one bits carry one transition each, so that clock data recovery is fairly easy with this coding, provided the signal doesn't consist of all zeros. The signals also carry significant DC content, which is not a factor in optics, though.

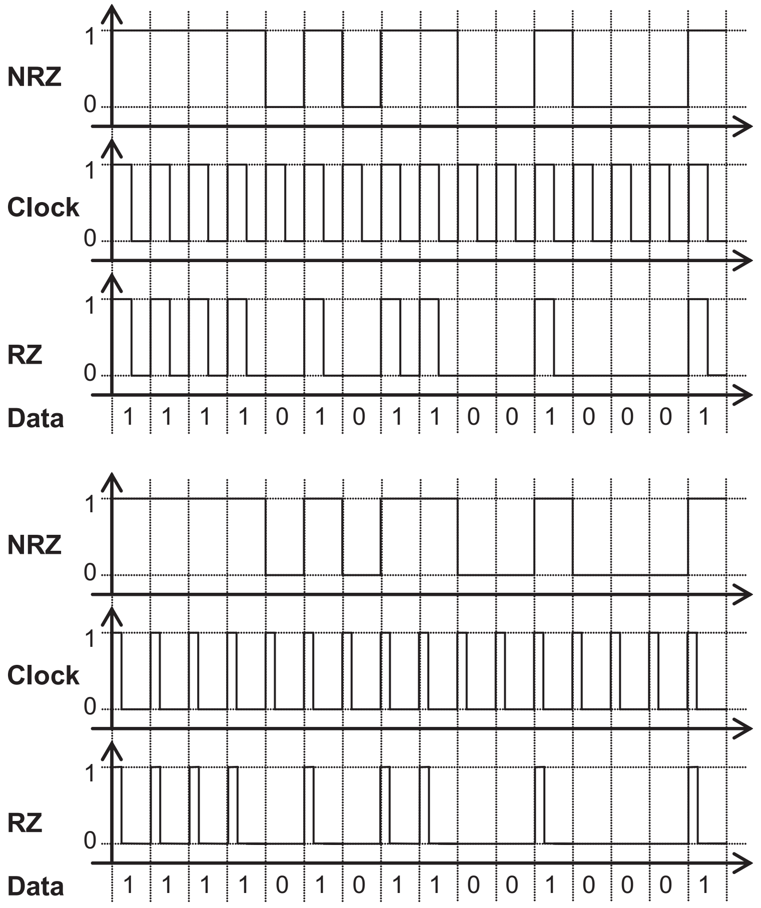


Figure 1–17 RZ coding of a short data sequence (PRBS 2^4-1). Top: 50% duty cycle. Bottom: 25% duty cycle.

The signal bandwidth of RZ-coded data is significantly higher than that of NRZ data, by at least a factor of two (for a 50% duty cycle). The spectral densities for the RZ-coded signals from Figure 1–17 are shown in Figure 1–18. The signal with a 50% duty cycle has significantly less energy at lower frequencies than the NRZ signal and very distinct spikes at the data rate and its even harmonics. The 25% duty cycle signal has even less low-frequency content but distinct spikes at all integer multiples of the data rate.

1.3.2.3 Return-to-One Code

Return-to-one (R1) code uses a static high level for the logical one state and a short low-level pulse for a zero. Creating an R1-formatted signal from NRZ data is a bit more complicated than using the RZ format: It's a binary AND of the inverted NRZ data with the clock, and the result

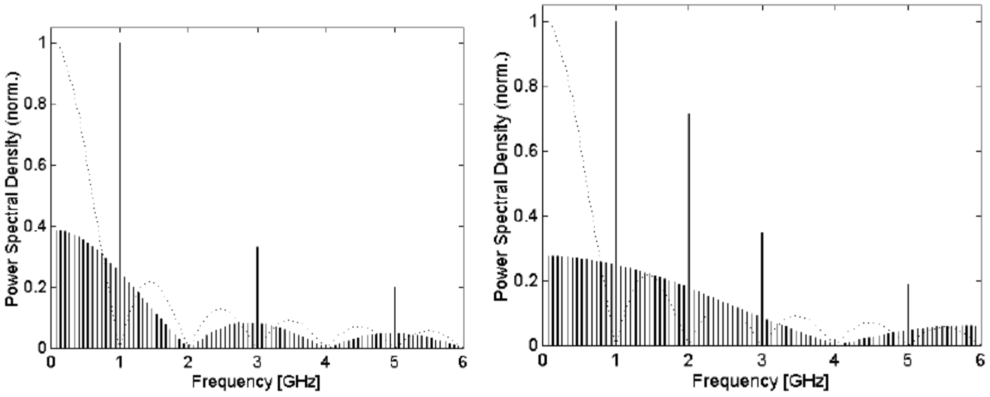


Figure 1–18 Power spectral density for a short RZ-formatted data sequence (PRBS 2^4-1), at 1 Gbit/s. Left: 50% duty cycle. Right: 25% duty cycle. Power density is normalized for comparison with NRZ format (dotted line).

inverted again. Figure 1–19 shows an example. The properties of R1-coded data are very similar to those of RZ-coded data, with the exception of the DC content, which is significantly higher than for RZ-coded signals.

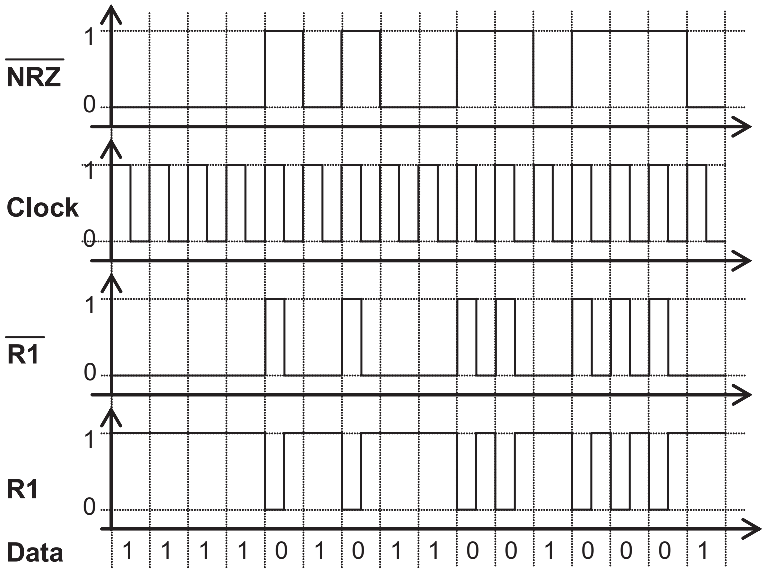


Figure 1–19 R1 coding of a short data sequence (PRBS 2^4-1)

1.3.2.4 Manchester Code

Manchester code is generated from NRZ data by a binary XOR with a clock signal. Since there are two possible clock phases, there are also two variants of Manchester code. The coded data has a transition in the middle of every bit, and the direction of this transition indicates a binary zero or one. The original Manchester variant uses a falling edge for a one and a rising edge for a zero; the other variant (which is used in IEEE 802.3 10Base-T Ethernet, for example) is the exact inverse. Figure 1–20 shows both variants.

Manchester code is very attractive for embedded clock applications because it forces at least one transition per bit, even if the data is a constant zero or one. It is also a DC-balanced code. However, the price for this is a significantly higher bandwidth relative to NRZ data. Figure 1–21 shows the spectral densities for two short data sequences. Compared to the NRZ spectrum

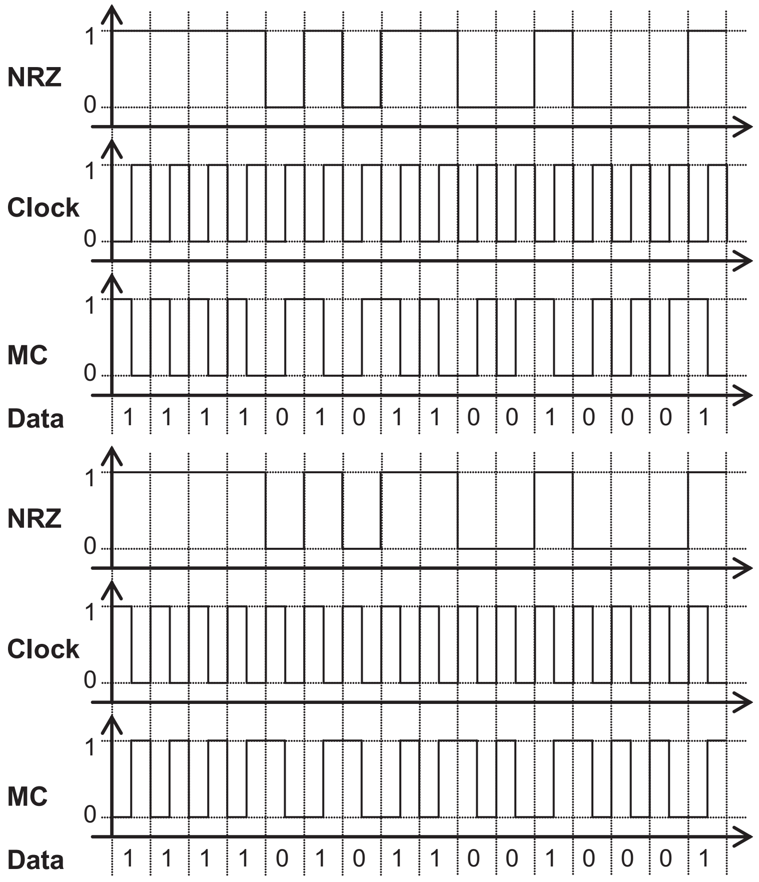


Figure 1–20 Manchester code representation of a short data sequence (PRBS 2^4-1). Top: “10” variant. Bottom: “01” variant.

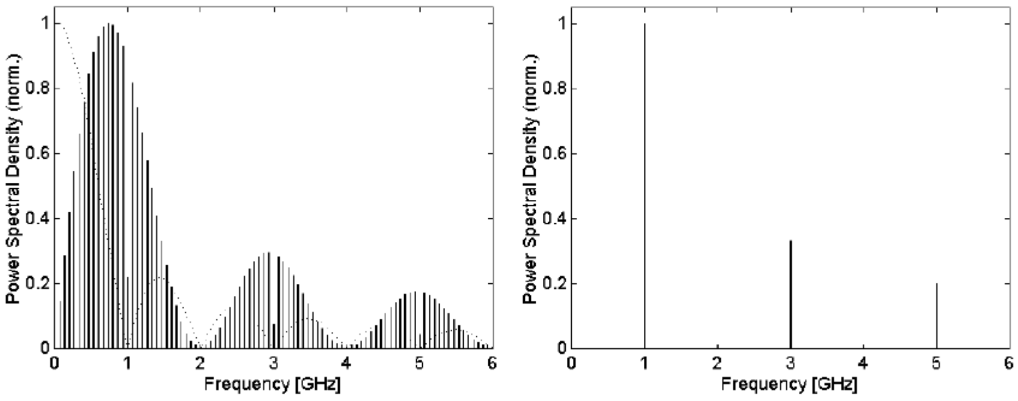


Figure 1–21 Power spectral density for Manchester-coded data at 1 Gbit/s. Left: PRBS 2^4-1. Right: Constant one (111111 . . .). Power density is normalized for comparison with NRZ format (dotted line, left plot only).

(dotted line), the PRBS spectrum has significantly less spectral content at low frequencies but more at higher frequencies. Spectral nulls are at even harmonics. The spectrum for the constant one pattern is equal to a 1 GHz square wave.

1.3.2.5 Non-Return-to-Zero Inverted Code

Non-return-to-zero inverted (NRZI) code is not, as the name suggests, the mere inversion of an NRZ-coded signal; it is an example of a differential code, where the state of the signal depends on both the current and the previous bit. An NRZI-coded signal changes its state when the current bit is a logic one bit but stays constant if the current bit is a logic zero (Figure 1–22). Using transitions rather than levels makes detection less error-prone in noise environments, and the signal polarity is insignificant. NRZI coding is used, for example, in USB.

The signal properties of NRZI-coded data are similar to those of NRZ data: The transition density can be between 0.0 (for a constant zero pattern) and 1.0 (for a constant one pattern), and

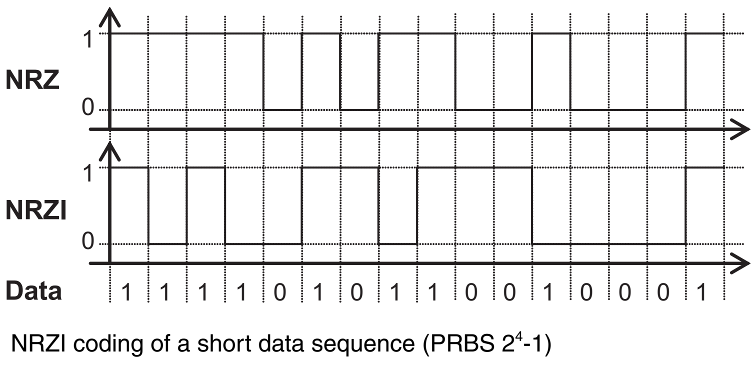


Figure 1–22 NRZI coding of a short data sequence (PRBS 2^4-1)

the spectral content for random data is exactly the same as for NRZ. The NRZI code is therefore not sufficient to enable data transmission with clock recovery, or to limit the amount of ISI.

1.3.2.6 Differential Manchester Code

Differential Manchester code (DMC) is a combination of Manchester and NRZI: It uses transitions in the middle of the bit, but the transition direction changes with every one in the data stream (Figure 1–23). This coding can be generated by an XOR function of NRZI-coded data and a clock signal. DMC is also known as conditional de-phase (CDP) code and used in token ring LANs (IEEE 802.5).

The properties of data that is coded with DMC are very similar to those of pure Manchester code: The signal is DC balanced, there is at least one transition per bit, and the spectrum has low content at lower frequencies but significantly more high-frequency content than NRZ data has.

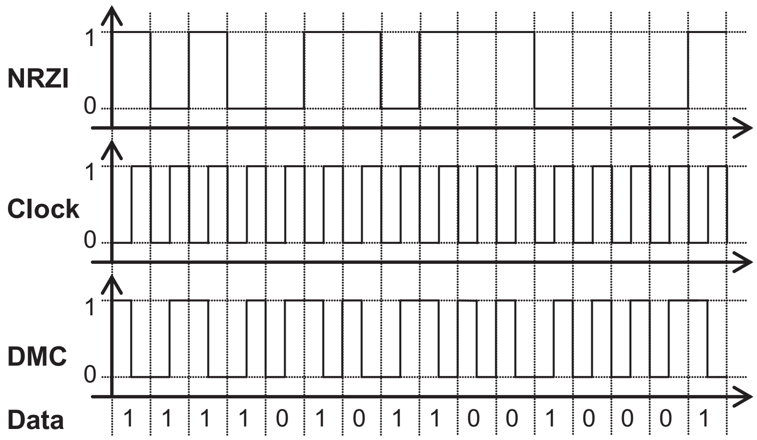


Figure 1–23 Differential Manchester coding of a short data sequence (PRBS 2^4-1)

1.3.3 Multilevel Line Codes

1.3.3.1 Bipolar Return-to-Zero Code

A variant of the RZ code is bipolar return-to-zero (BPRZ) coding, where the signal returns to an intermediate zero level after both zero and one bits (Figure 1–24). There are two transitions per bit, which makes synchronization of the receiver fairly easy. The drawback is the fairly complicated circuitry and an even higher bandwidth requirement than for RZ and R1 data. Figure 1–25 shows the power spectral density for a BPRZ-formatted data sequence.

1.3.3.2 Pulse Amplitude Modulation

Pulse amplitude modulation (PAM) is a class of multilevel codes that encodes several consecutive bits into one of several levels. PAM-4, for example, encodes two bits into one out of four levels (Figure 1–26). Demodulation is performed by detecting the signal level once per symbol

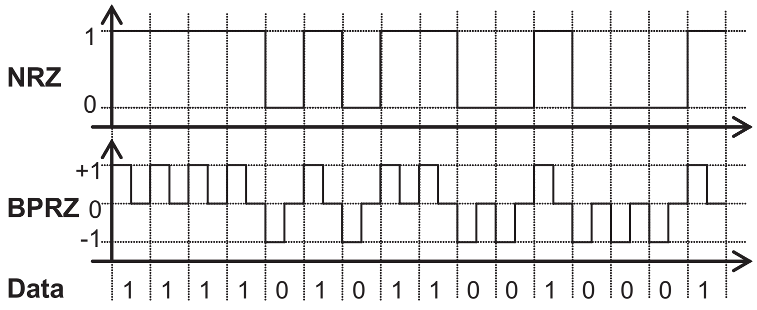


Figure 1–24 BPRZ coding of a short data sequence (PRBS 2^4-1)

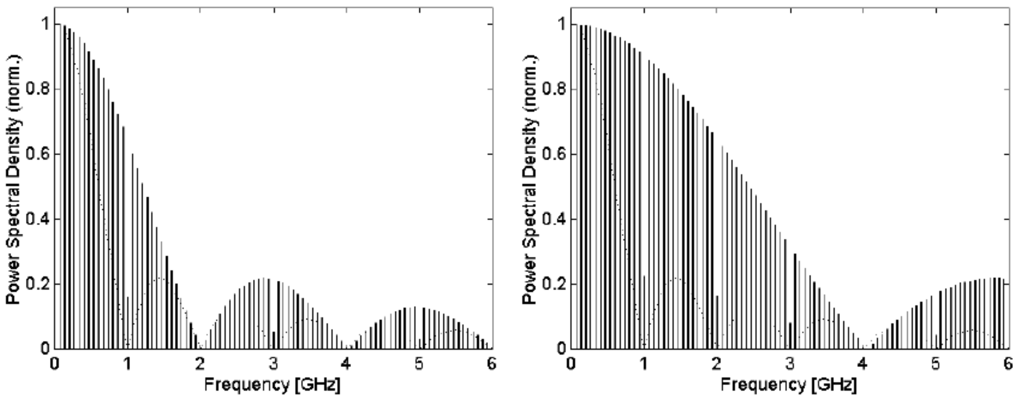


Figure 1–25 Power spectral density for a short BPRZ-formatted data sequence (PRBS 2^4-1), at 1 Gbit/s. Left: 50% duty cycle. Right: 25% duty cycle. Power density is normalized for comparison with NRZ format (dotted line).

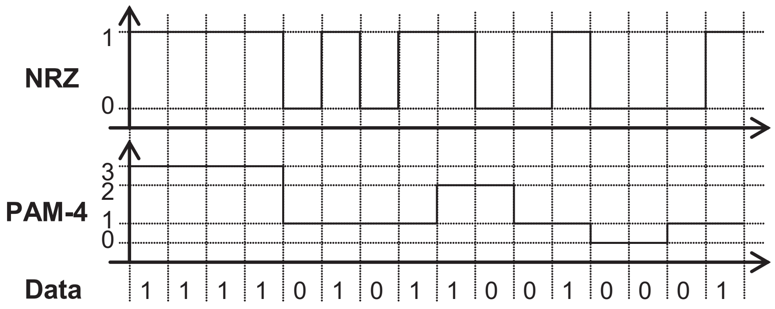


Figure 1–26 PAM-4 coding of a short data sequence (PRBS 2^4-1)

period. PAM-4-encoded data has much less high-frequency content than, for example, NRZ data because the signal level changes only for every other bit. However, the cost is increased transmitter and especially receiver complexity, and a lower signal-to-noise ratio if the same levels are used. PAM-4 alone is not sufficient for embedded clock systems, as it does not guarantee transition density: Constant zero or one patterns are encoded as DC levels. Figure 1–27 shows the power spectral density for a PAM-4-coded data sequence.

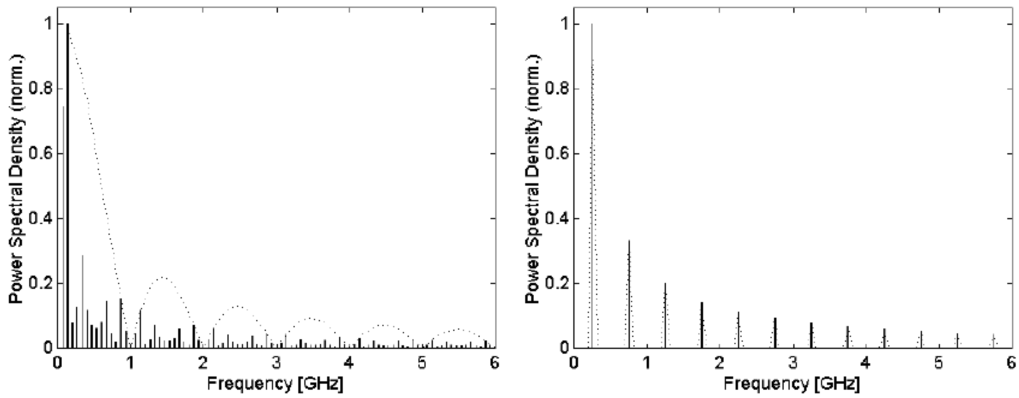


Figure 1–27 Power spectral density for PAM-4-coded data at 1 Gbit/s. Left: PRBS 2^4-1. Right: Half-rate toggle (11001100 . . .). Power density is normalized for comparison with NRZ format (dotted line).

1.3.4 Block Codes

1.3.4.1 mBnB Block Codes

Block codes of type mBnB take m bits of the original data and encode them into n bits, following very specific rules. Several of the coding schemes from the previous sections can be expressed as 1B2B codes; RZ coding, for example, encodes every one bit as a one, followed by a zero, and every zero bit as two zeros. Widely used in serial high-speed applications are 4B5B and in particular 8B10B coding. The dominant encoding scheme in computing applications, 8B10B seems to hit a sweet spot with relatively low overhead (25%), ease of implementation, coding properties such as maximum run length, and so on. Chapter 3 describes 4B5B and 8B10B coding in greater detail.

1.3.4.2 Error Detection and Forward Error Correction

Some of the block codes from Section 1.3.4.1 enable the receiver to detect some transmission errors, either from calculating disparity or by detecting invalid code words. A system that is based on such coding techniques can issue a packet resend command and transmit the packet again, this time hopefully without an error. Ideally, however, the receiver would be able to not only detect errors (all errors, not just a few) but also correct them.

The process of adding redundancy to the data stream and analyzing and correcting errors in real time is called forward error correction (FEC). Systems that use FEC can operate with less margin in transmission than non-FEC systems. In practical applications, this means a longer range between sender and receiver or reduced transmission power. Especially under difficult transmission conditions, FEC systems are more effective than non-FEC systems because fewer packets need to be retransmitted.

1.4 Electrical Signaling

1.4.1 Single-Ended Signaling

Single-ended systems use a shared reference rail for both the transmitter and the receiver that provides the reference level for zero/one bit decisions and also carries the return current (Figure 1–28). The reference rail (or ground) needs to be sufficiently low in impedance; otherwise, the return currents will cause a voltage drop that reduces the signal across the receiver inputs. Other concerns are noise coupling and electromagnetic interference.

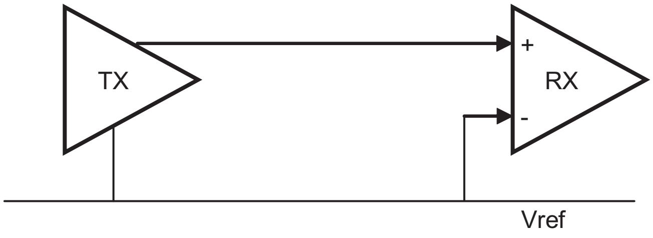


Figure 1–28 Single-ended system. The receiver makes bit decisions based on the difference between the signal and a reference voltage.

1.4.2 Differential Signaling

Differential signaling uses pairs of wires: One of the wires carries the signal, while the other wire carries the inverse of the signal (Figure 1–29). The receiver makes its bit decisions based on

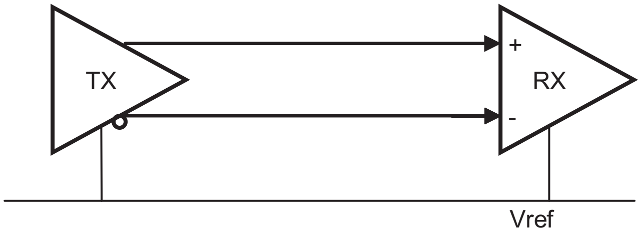


Figure 1–29 Differential system. The receiver makes bit decisions based on the difference between two signals; the reference voltage is irrelevant.

the difference between the two wires and thereby removes the dependency on the signal ground; a differential signal essentially carries its own reference.

We call the two signals signal A and signal B. The differential signal is then the difference between the two signals:

$$V_{diff}(t) = V_A(t) - V_B(t) \qquad \text{(Equation 1–14)}$$

The common signal is the average of the two signals:

$$V_{common}(t) = \frac{V_A(t) + V_B(t)}{2} \qquad \text{(Equation 1–15)}$$

Figure 1–30 shows an example. The two single-ended signals with 400 mV amplitude (peak to peak) and an offset of 1.0 V can be decomposed into an 800 mV peak-to-peak differential signal and a 1.0 V constant common signal.

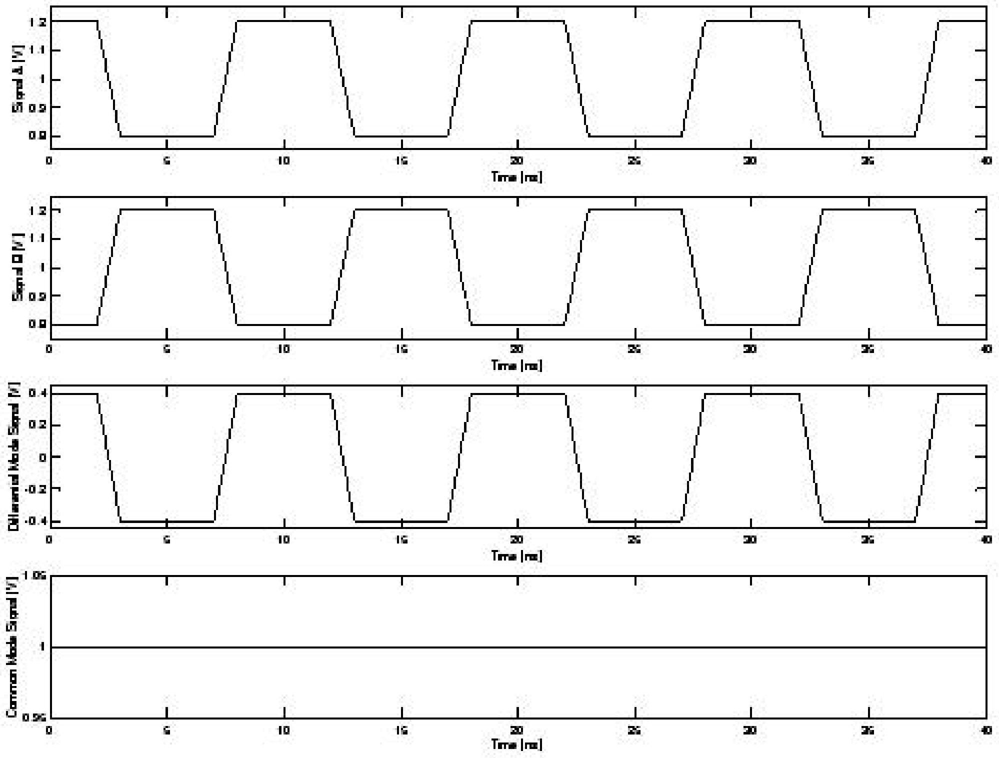


Figure 1–30 Differential signaling example: 100 MHz digital clock signal, single-ended amplitude of 400 mV (peak to peak) with a 1.0 V offset. Top to bottom: Signal A, signal B, differential signal, common signal.

A differential system is fully described either by the two signals A and B or by the differential and common signals. If the differential and common signals are known, the signals on the two lines can be recovered as the sum (for V_A) and difference (for V_B) of the common signal and half the differential signal:

$$V_A(t) = V_{common}(t) + \tfrac{1}{2}V_{diff}(t) \qquad \textbf{(Equation 1–16)}$$

$$V_B(t) = V_{common}(t) - \tfrac{1}{2}V_{diff}(t) \qquad \textbf{(Equation 1–17)}$$

If the two wires are routed in parallel, they receive nearly the same interference; this makes the system almost immune to most types of common mode noise coupling and electromagnetic interference. Figure 1–31 shows an example. Note how the sinusoidal interference signal shows up in the common mode of the signal, while the differential signal is not affected at all.

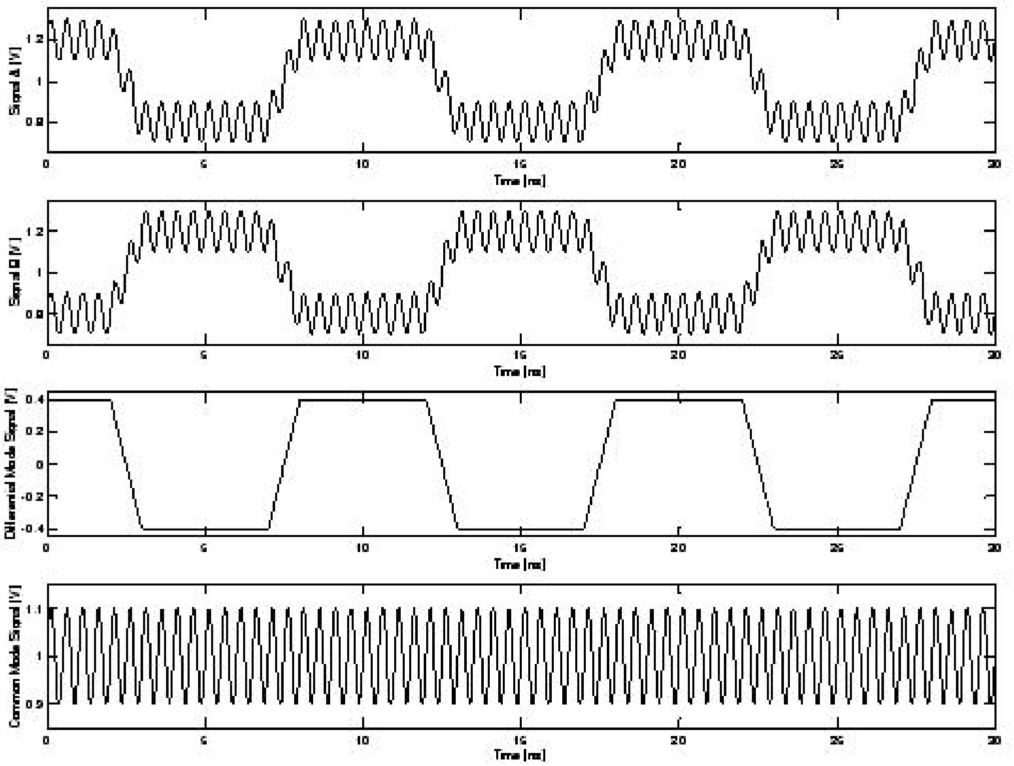


Figure 1–31 Differential signaling example: 100 MHz digital clock signal, single-ended amplitude of 400 mV (peak to peak) with a 1.0 V offset. Interference signal is a 5 GHz sinusoidal, with amplitude of 200 mV peak to peak. Top to bottom: Signal A, signal B, differential signal, common signal.

Because of its noise immunity, differential signaling is the method of choice for almost all high-speed serial data transmission systems. There are associated costs, however: Two wires per signal require twice as many pins and double the routing space. Because differential pairs need to be routed close together, and ideally with low skew, the routing complexity increases. However, this additional complexity is offset because we don't need nearly as many ground signals as for single-ended signaling.

1.4.3 Preemphasis and Receiver Equalization

High-speed serial signaling suffers from bandwidth limitations in transmission channels. Especially in cost-sensitive applications, where less expensive board materials are used, signal distortions introduced by the channel can render a signal unusable.

Signal distortions due to high-frequency cutoff can be moderated to some degree by signal-processing techniques in the transmitter. Two options are available: We can either amplify the higher-frequency spectral components of the signal or attenuate the low-frequency content of the signal. The former is called preemphasis, the latter deemphasis. Signal-processing methods can also be applied to the receiver. Several types of receiver equalization can extract a meaningful signal out of an apparently random signal.

1.5 Summary

In this chapter, we've introduced the basic concepts of high-speed serial transmission systems: clock architectures, line coding, and differential electrical signaling with preemphasis or receiver equalization.

The remainder of this book describes how we can characterize such a system, either as a whole or individually for its components. Transmitter tests verify the electrical performance of the signal before it enters the channel, while receiver tests verify that the worst-case realistic signals can be understood by the receiver. Channel tests finally determine the quality of the transmission medium.

1.6 References

1.6.1 General References

1. C. E. Shannon. "A Mathematical Theory of Communication." *Bell System Technical Journal* 27 (July, October 1948): 379–423, 623–656. [This classic paper is available from the Bell Web site at http://plan9.bell-labs.com/cm/ms/what/shannonday/paper.html.]

2. Hubert Zimmermann. "OSI Reference Model: The ISO Model of Architecture for Open Systems Interconnection." *IEEE Transactions on Communications* 28, no. 4 (1980): 425–432.

3. SDRAM Specification, JESD79-2B. Arlington, VA: JEDEC Solid State Technology Association, January 2005.

4. Peripheral Component Interconnect (PCI) Specification, Revision 3.0. Beaverton, OR: PCI Special Interest Group, 2004.

1.6.2 Digital Design

5. Stephen Hall, Garrett Hall, and James McCall. *High-Speed Digital System Design*. New York: Wiley, 2000.

6. Howard Johnson and Martin Graham. *High-Speed Digital Design: A Handbook of Black Magic*. Englewood Cliffs, NJ: Prentice Hall, 1993.

7. Howard Johnson and Martin Graham. *High-Speed Signal Propagation: Advanced Black Magic*. Upper Saddle River, NJ: Prentice Hall, 2003.

8. Eric Bogatin. *Signal Integrity: Simplified*. Upper Saddle River, NJ: Prentice Hall, 2004.

9. Yoshitaka Takasaki. *Digital Transmission Design and Jitter Analysis*. Norwood, MA: Artech House, 1991.

1.6.3 Line Coding

10. A. Lender. "The Duobinary Technique for High Speed Data Transmission." *IEEE Transactions on Communication and Electronics* 82 (May 1963): 214–218.

11. Andrew Sekey. "An Analysis of the Duobinary Technique." *IEEE Transactions on Communication Technology* Com-14, no. 2 (1966): 126–130.

12. Jeffrey H. Sinsky, Marcus Duelk, and Andrew Adamiecki. "High-Speed Electrical Backplane Transmission Using Duobinary Signaling." *IEEE Transactions on Microwave Theory and Techniques* 53, no. 1 (2005): 152–160.

13. R. Forster. "Manchester Encoding: Opposing Definitions Resolved." *IEEE Engineering Science and Education Journal* 9, no. 6 (2000): 278–280.

14. X. Widmer and P. A. Franaszek. "A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code." *IBM Journal of Research and Development* 27, no. 5 (1983): 440–451.

15. Alex Deas, David R. Stauffer, and Anthony Sanders. "No 'Magic Bullet' for Signaling Schemes." *EE Times,* April 19, 2004.

1.6.4 Forward Error Correction

16. Irving S. Reed and Gustave Solomon. "Polynomial Codes Over Certain Finite Fields." *Journal of the Society for Industrial and Applied Mathematics* 8, no. 2 (1960): 300–304.

17. S. Lin and D. J. Costello. *Error Control Coding*. Englewood Cliffs, NJ: Prentice Hall, 1982.

18. A. M. Michelson and A. H. Levesque. *Error Control Techniques for Digital Communication*. New York: Wiley, 1985.

19. W. W. Peterson and E. J. Weldon, Jr. *Error Correcting Codes*, 2nd ed. Cambridge, MA: The MIT Press, 1972.

20. V. Pless. *Introduction to the Theory of Error-Correcting Codes*, 3rd ed. New York: Wiley, 1998.

21. C. Schlegel and L. Perez. *Trellis Coding*. Piscataway, NJ: IEEE Press, 1997.

22. S. B. Wicker. *Error Control Systems for Digital Communication and Storage*. Upper Saddle River, NJ: Prentice Hall, 1995.

1.6.5 Equalization and Preemphasis

23. Jin Liu and Xiaofeng Lin. "Equalization in High-Speed Communications Systems." *IEEE Circuits and Systems Magazine* 4, no. 2 (2004): 4–17.

24. A. Sanders, M. Resso, and J. D'Ambrosia. "Channel Compliance Testing, Utilizing Novel Statistical Eye Methodology." DesignCon, Santa Clara, CA, January 2004.