DesignCon 2004

SerDes Architectures and Applications

Dave Lewis, National Semiconductor Corporation

Abstract

When most system designers look at serializer/deserializer (SerDes) devices, they often compare speed and power without considering how the SerDes works and what it actually does with their data. Internal SerDes architecture may seem irrelevant, but this overlooked item can dictate many important system parameters like system topology, protocol overhead, data formatting and flow, latency, clocking and timing requirements, and the need for additional buffering as well as logic. These issues can have a big impact on system cost, performance, and efficiency.

There are at least four distinct SerDes architectures. They include: parallel clock SerDes, 8b/10 SerDes, embedded clock bits (alias start-stop bit) SerDes, and bit interleaving SerDes. Each one has evolved over the years to address a certain set of system design issues. This paper unveils the inner workings of these four SerDes architectures, examines their differences, and shows how each fits an important range of today's applications.

Author(s) Biography

Dave Lewis is a Technical Marketing Manager in National Semiconductor's PC & Networking Group, handling high-speed interface products. He is the author of many articles and design guides including the original "LVDS Owner's Manual." He holds a BSEE from the University of California at San Diego.

Introduction

Serial interconnects form the critical backbone of modern communications systems, so the choice of serializer/deserializer (SerDes) can have a big impact on system cost and performance. While the maze of choices may seem confusing at first, SerDes devices fall into a few basic architectures, each tailored to specific application requirements. A basic understanding of these architectural differences enables the designer to quickly find the right SerDes for the application. In this article we examine four distinct SerDes architectures and show how each plays a vital role in today's systems.

SerDes Architectures

Parallel Clock SerDes

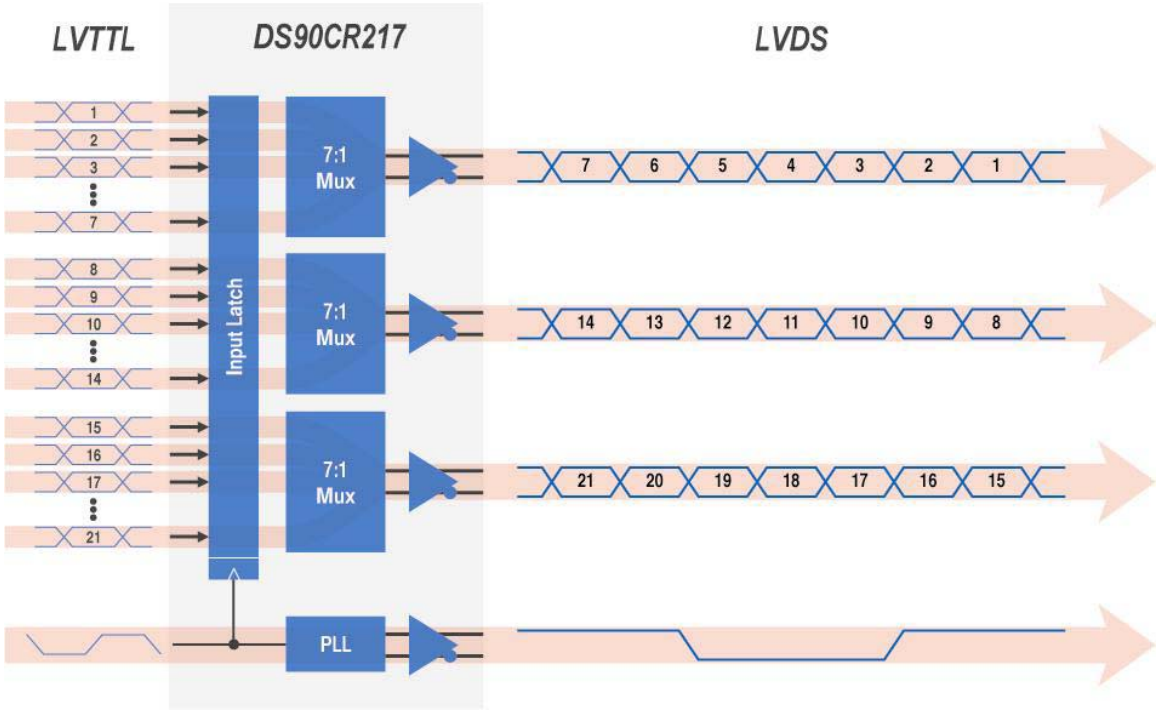


Figure 1. Parallel clock serializer coding example.

Parallel clock SerDes are normally used to serialize wide "data-address-control" parallel buses such as PCI, UTOPIA, processor buses, and control buses, etc. Instead of tackling the whole bus with one multiplexer, the parallel clock SerDes architecture employs a bank of n-to-1 multiplexers, each serializing its section of the bus separately. The resulting serial data streams travel to the receiver in parallel with an additional clock signal pair that the receiver uses to latch in and recover the data. Since clock and data travel on multiple pairs, pair-to-pair skew must be minimized for proper deserialization.

Embedded Clock (Start-Stop) Bits SerDes

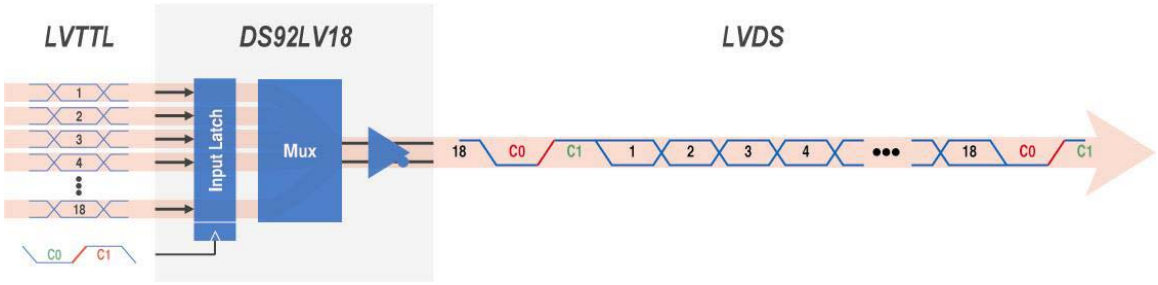


Figure 2. 18-bit embedded clock bits serializer coding example.

The embedded clock bits architecture transmitter serializes the data bus and the clock onto one serial signal pair. Two clock bits, one low and one high, are embedded into the serial stream every cycle, framing the start and end of each serialized word (hence the alternative name "start-stop bit" SerDes) and creating a periodic rising edge in the serial stream. Data payload word widths are not constrained to byte multiples; 10- and 18- bit widths are popular bus widths.

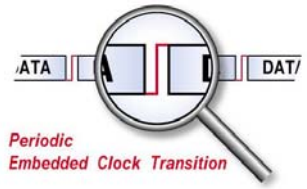
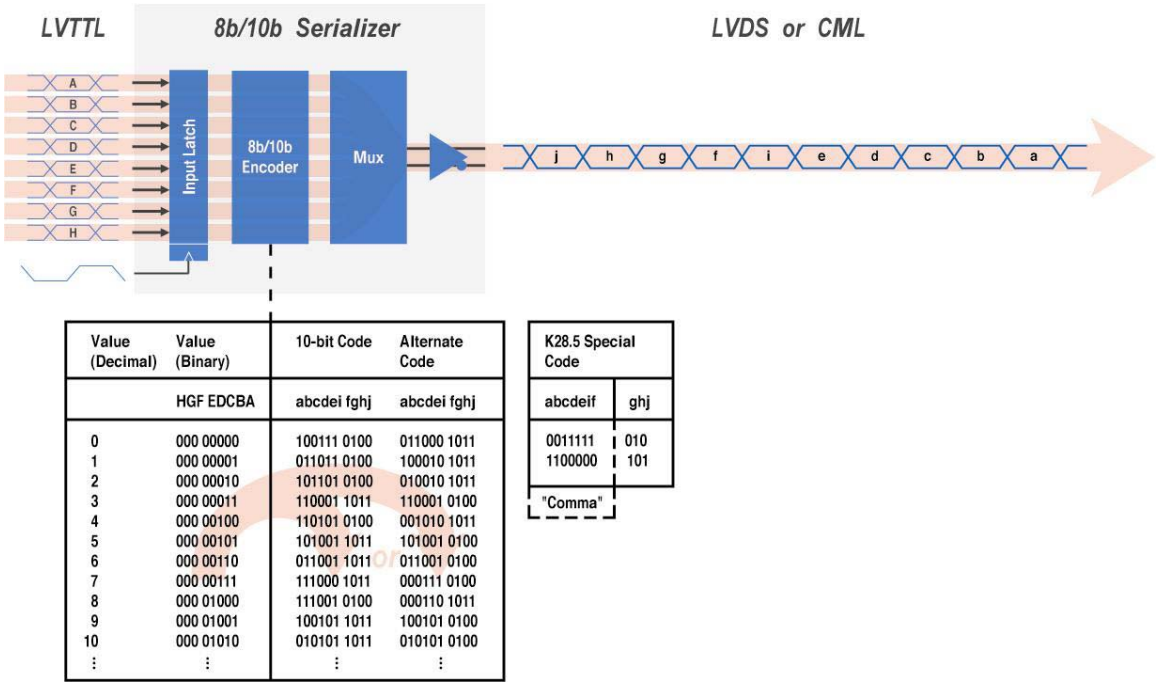


Figure 3. Periodic embedded clock transition.

After powering up, the receiver automatically searches for the periodic embedded clock rising edge. Since the data payload bits change value over time while the clock bits do not, the receiver is able to locate the unique clock edge and synchronize to it. Once locked, the receiver recovers data from the serial stream regardless of payload data pattern. This automatic synchronization capability is commonly called "lock to random data" and requires no external system intervention. This is an especially useful feature in systems where the receiver is in a remote module not under direct system control. Since the receiver is locked to the incoming embedded clock and not an external reference clock, jitter requirements for both transmitter and receiver input clocks are relaxed significantly.

8b/10b SerDes



Value (Decimal)	Value (Binary)	10-bit Code	Alternate Code
	HGF EDCBA	abcdei fghj	abcdei fghj
0	000 00000	100111 0100	011000 1011
1	000 00001	011011 0100	100010 1011
2	000 00010	101101 0100	010010 1011
3	000 00011	110001 1011	110001 0100
4	000 00100	110101 0100	001010 1011
5	000 00101	101001 1011	101001 0100
6	000 00110	011001 1011	011001 0100
7	000 00111	111000 1011	000111 0100
8	000 01000	111001 0100	000110 1011
9	000 01001	100101 1011	100101 0100
10	000 01010	010101 1011	010101 0100
⋮	⋮	⋮	⋮

K28.5 Special Code	
abcdeif	ghj
0011111	010
1100000	101
"Comma"	

Figure 4. 8b/10b serializer coding example.

The 8-bit/10-bit (8b/10b) serializer maps each parallel data byte to a 10-bit code and serializes the 10-bit code onto a serial pair. The 10-bit transmission codes were developed by IBM Corporation[1] in the early 1980's and guarantee both multiple edge transitions every cycle as well as DC balance (balanced number of transmitted ones and zeros). Frequent edge transitions in the stream allow the receiver to synchronize to the incoming data stream. DC balance facilitates driving AC-coupled loads, long cables and optical modules.

In order for the receiver to locate the 10-bit code word boundaries in the serial stream, the transmitter first marks one such boundary by sending a special symbol called a comma character. The unique bit sequence in this comma character never appears in normal data traffic and acts as reliable a marker for receiver code alignment. Once code alignment is accomplished, the receiver maps the 10-bit codes back to byte data, flagging an error if it detects an invalid 10b code.

Most 8b/10b deserializer architectures monitor lock by comparing the recovered clock frequency to an external reference clock. As a result, they typically require tight external clock source frequency and jitter control.

Bit Interleaving SerDes

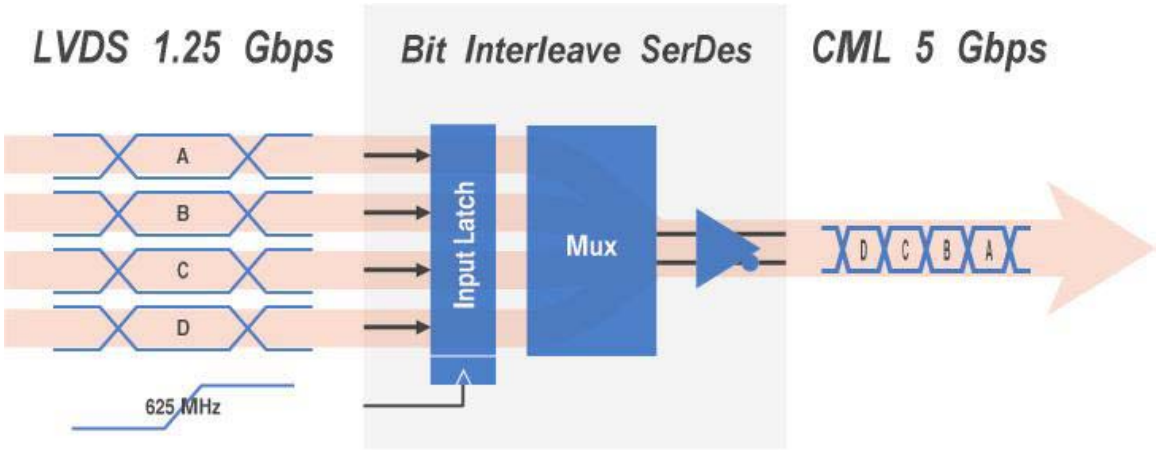


Figure 5. Bit interleaving serializer coding example.

Bit interleaving SerDes multiplex several slower SONET/SDH or 8b/10b serial streams into one faster serial stream by interleaving the bits. The receiver demultiplexes the bits back into the original slower streams. Note that a serial stream coming into transmitter input channel 1 may not come out on receiver output channel 1. This is not regarded as a problem in the applications because the serial streams contain independent cell or packet data that is processed downstream. Due to their high-speed nature and low jitter requirements, bit interleaving SerDes require very precise external clocks.

Applications

Parallel Clock SerDes

Parallel clock SerDes are normally used to serialize traditional wide "data+address+control" buses, acting as a "virtual ribbon cable" unidirectional bridge.

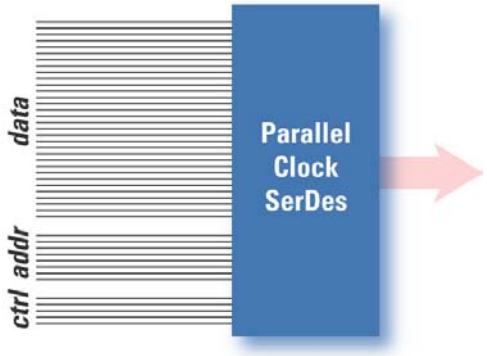


Figure 6. Parallel clock SerDes accommodate traditional wide parallel buses with address and control as well as data signals.

Despite requiring multiple serial pairs, parallel clock SerDes still deliver benefits over non-serialization such as fewer wires (especially grounds), lower power, longer cable driving capability, lower noise/EMI, and lower cable/connector costs. Not being confined to using one serial pair, parallel clock SerDes can be made arbitrarily wide and also avoid the design issues associated with ultra high speed serial data rates. Parallel clock SerDes offer excellent price/performance and are often the only practical way to transmit a traditional wide parallel bus over several meters of cable. Common parallel bus widths for these chipsets include 21-, 28-, and 48- bits.

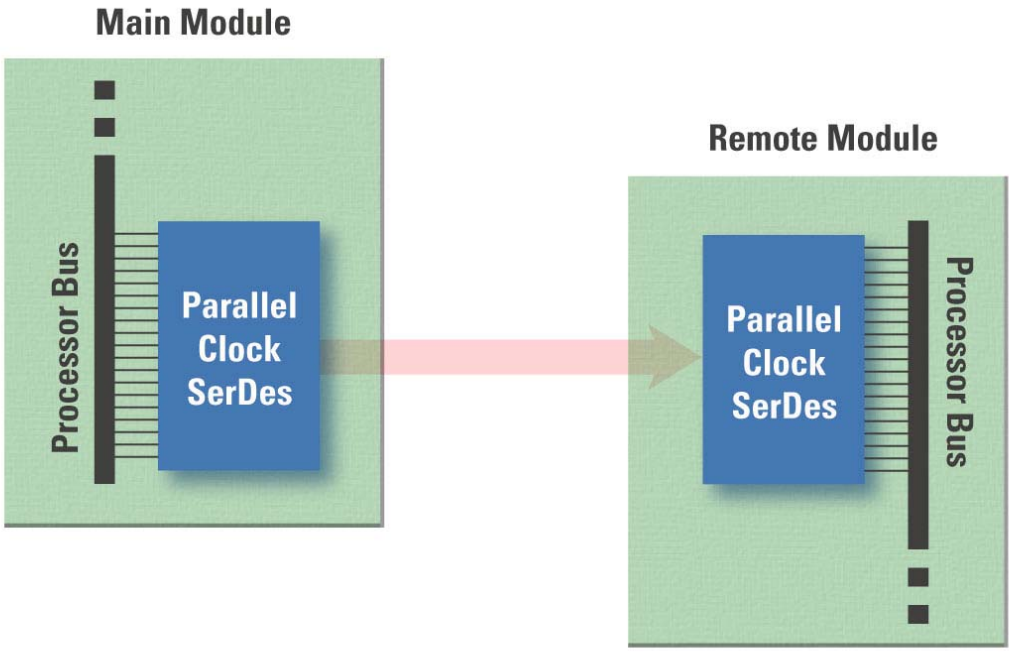
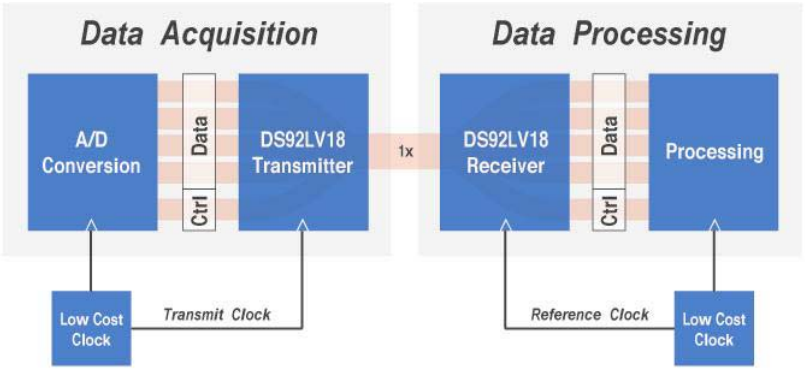


Figure 7. Unidirectional rack-to-rack processor bus extension.

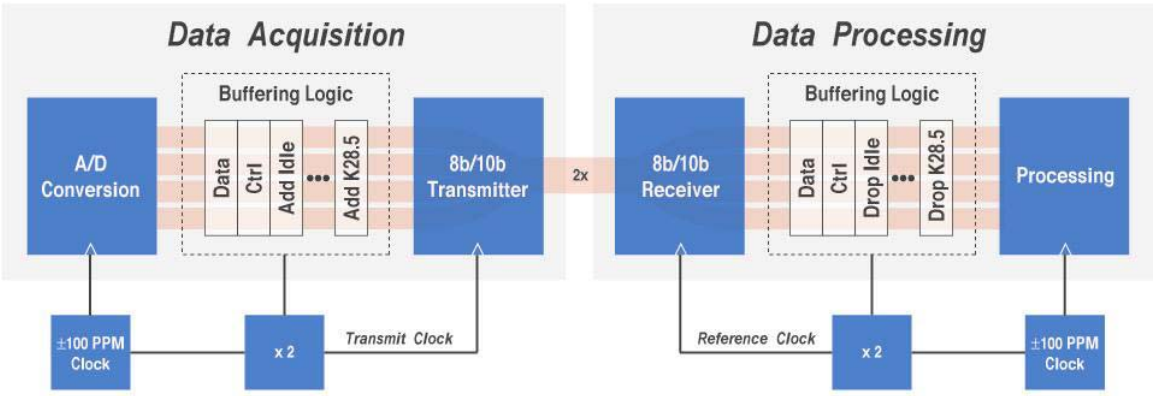
Common applications include stackable Ethernet switch expansion, rack-to-rack and shelf-to-shelf datacom/telecom interconnect, and video/camera links.

Embedded Clock (Start-Stop) Bits SerDes

Embedded clock bits SerDes are especially well suited to applications that transmit raw data plus other signals such as control, parity, frame, sync, status, etc. An application example for serializing 18 bits is shown below. The 18-bit transmitter serializes not only the data but also two extra bits of additional information such as parity and frame. These bits are serialized along with the data at the normal A/D sampling rate so no data buffering or extra logic is required.



Signal processing system: DS92LV18 SerDes implementation example.



Signal processing system: 8b/10b SerDes implementation example.

Figure 8. Signal processing system example implementations based on DS92LV18 SerDes (above) and 8b/10b SerDes (below).

Using a byte-oriented 8b/10b SerDes in the same application would be more complicated. The extra non-byte-oriented control information must be buffered and sent in byte format. A K28.5 comma character must also be sent at the start of link synchronization, requiring additional logic. These extra “non-data” bytes require the SerDes to operate faster than the data conversion rate, placing higher demands on backplane or cable design and also requiring some kind of idle insertion/deletion flow control mechanism. While in data communications systems such buffering typically already exists, in many non-datacom applications this extra processing and buffering must be added.

Another feature of the embedded clock bits SerDes is automatic receiver lock to random data. This is an especially useful feature in systems where the receiver is in a remote module not under direct system control and also in systems where one transmitter broadcasts to multiple receivers. In the broadcast case, a new receiver module inserted onto the bus will lock to random data without the need to interrupt traffic to the other receivers by sending training patterns or characters.

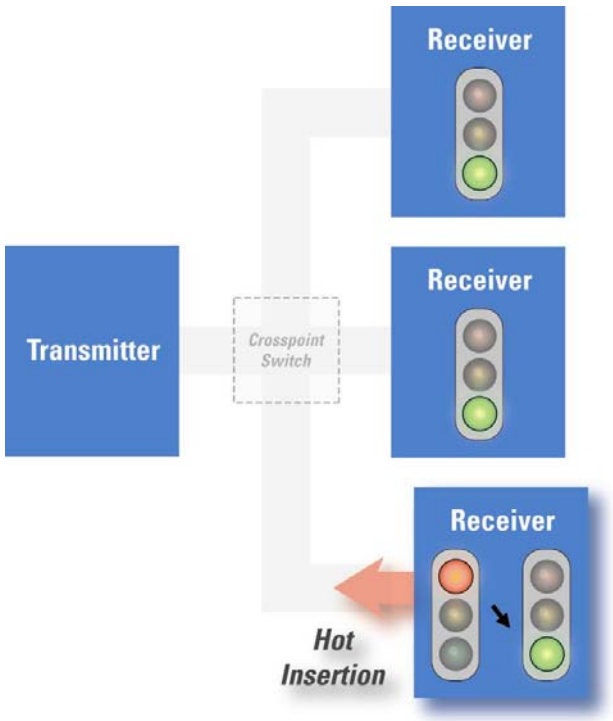


Figure 9. Automatic receiver lock to random data in a broadcast topology during hot insertion.

The embedded clock bits deserializer locks to and tracks the incoming embedded clock rising edge, requiring a reference clock only during initial synchronization to prevent lock to a false harmonic. This relaxes the jitter requirement on both transmit and reference clocks by at least an order of magnitude (see table below), reducing the cost of the clock oscillators and clock distribution networks. In many cases, an inexpensive PC-grade oscillator can be used to generate the receiver reference clock.

	Embedded Clock Bits SerDes	Other SerDes
Serializer Transmit Input Clock Jitter	80 or 120 ps RMS	5 or 10 ps RMS
Deserializer Reference Clock vs. Serializer Transmit Clock Disparity	± 50000 PPM	± 100 PPM

Figure 10. Comparison illustrating relaxed clocking requirements of embedded clock bits SerDes versus typical SerDes chipsets.

Embedded clock bits SerDes are well suited to non-byte-oriented applications such as applications requiring transmission of unpacketized raw data plus control signals. Examples include signal-processing systems such as base stations, automotive

imaging/video and sensor systems where an analog-to-digital converter, camera or display communicates raw data with the processing unit at the other end of the link.

8b/10b SerDes

8b/10b SerDes are well suited to serializing byte-oriented data such as cell or packet traffic across backplanes, cable and fiber. Many standards such as Ethernet, Fibre Channel, InfiniBand and others use the popular 8b/10b coding at rates of 1.0625, 1.25, 2.5, and 3.125 Gbps and many SerDes are available which span these data rates.

8b/10b coding has a maximum run length (the maximum number of consecutive ones or zeros in the serial stream) of 5 bits. This limits the spectral content of the serial stream that can ease the task of suppressing electromagnetic radiation. For example, given a 1 Gbps line rate after 8b/10b coding, the maximum and minimum 1st harmonic frequencies are 1 GHz and (1 GHz)/5 = 200 MHz. (The max and min fundamental frequencies are therefore 500 and 100 MHz, respectively.)

Minimizing the run length can also reduce deterministic jitter due to inter-symbol interference (ISI) on lossy interconnects, though this effect is small for most applications. ISI is caused by interconnects that attenuate higher frequencies more than low frequencies, "smearing" longer consecutive bits sequences into surrounding isolated bits.

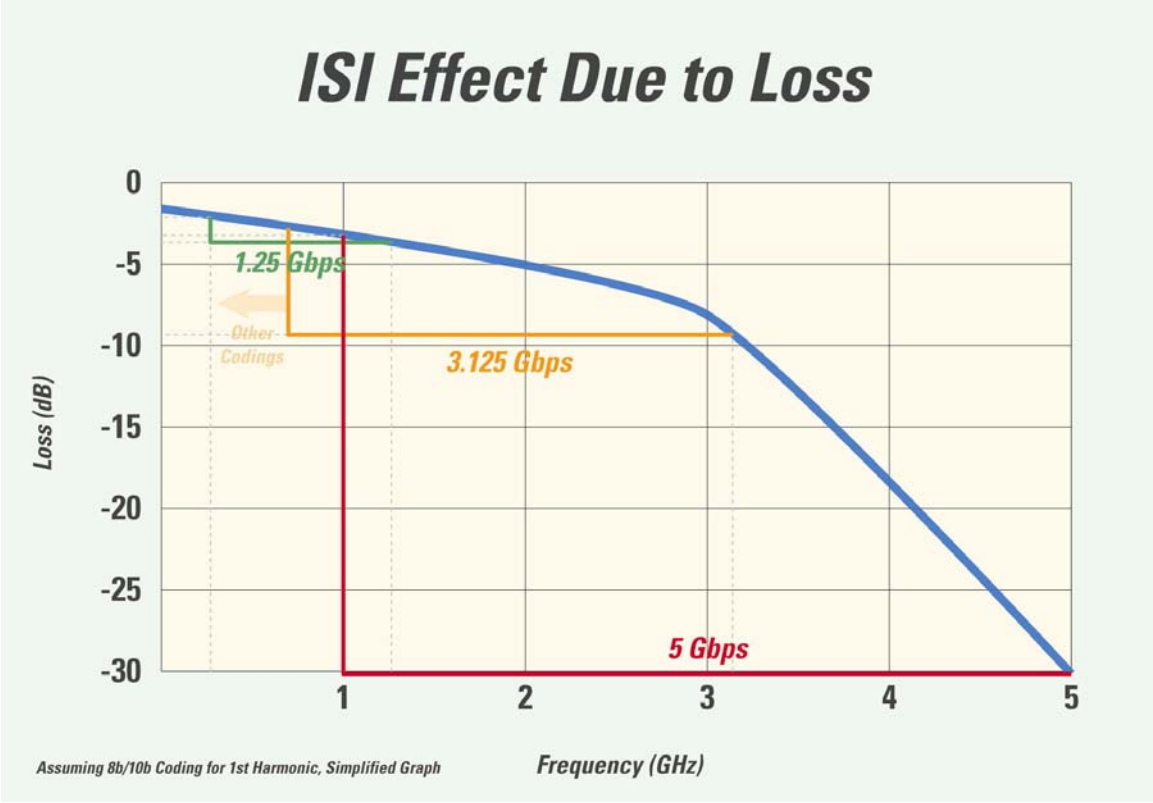


Figure 11. ISI versus signaling rate and run length for a typical FR4 backplane.

The graph above shows a typical FR4 backplane S21 loss versus frequency curve. Plotting the max and min frequencies for the 1st harmonics, we see the slope between the points—hence the ISI—increases dramatically with signaling rate due to the sharp roll off in loss above 3 GHz. Increasing the run length (and therefore decreasing the min frequency), however, does not have a large effect on ISI since the loss curve is relatively flat at low frequencies.

8b/10b serial streams are DC balanced, meaning the running disparity—or the number of ones sent minus the number of zeros sent—is on average equal to zero. 8b/10b data code words have a disparity of +2, 0, or –2, so the running disparity of an 8b/10b serial data stream always lies between +2 and –2.

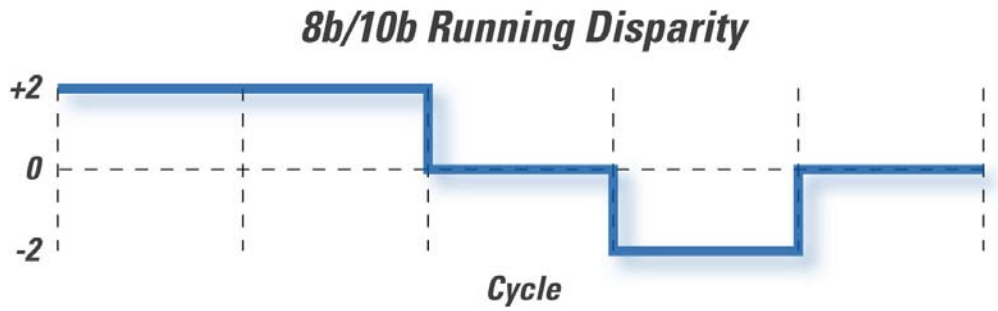


Figure 12. 8b/10b running disparity example.

DC balance coding as well as short run length are necessary for reliably driving AC coupled environments and fiber optic modules. This is a major advantage of 8b/10b coding as more serial interconnects go optical. In addition, DC balance can help extend cable drive capability.

8b/10b coding also provides a way to check errors and send control information. As described earlier, most of the possible 10-bit code permutations are not valid 8b/10b data code words. This allows 8b/10b deserializers to flag invalid codes and provide a level of error checking similar to using a parity bit. While this scheme does not count total bit errors, it is a good way to monitor serial link performance. In addition to data code words, many standards also define control words such as packet/frame markers, fault flags, alignment characters, etc. These control code words help systems assemble and disassemble packets, making 8b/10b coding very popular in communications data processing systems.

Bit Interleaving SerDes

Bit interleaving SerDes are commonly used in telecom transmission equipment such as add-drop multiplexers and pseudo-optical switches to aggregate SONET/SDH streams for transmission over cable or fiber to the core network. Common serializer configurations include 4 x 155 Mbps to 622 Mbps and 4 x 622 Mbps to 2.488 Gbps mux/demux functions.

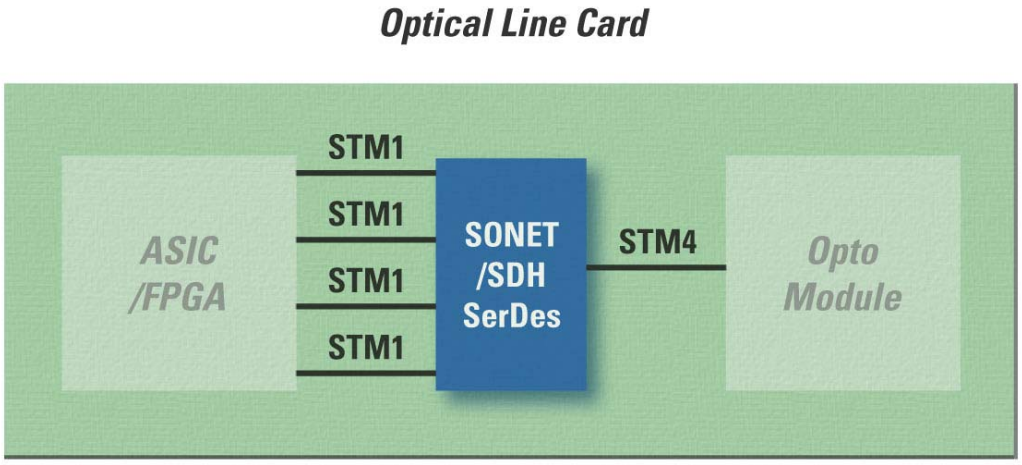


Figure13. Bit interleaving SerDes 4 x STM1 (155 Mbps) to 1 x STM4 (622 Mbps) application example.

Other types of bit interleaving SerDes chips multiplex 8b/10b-coded streams and are employed in switch and router equipment to get more bandwidth through existing backplanes. In the example below, a bit interleaving SerDes is used to multiplex up to 4 x 1.25 Gbps 8b/10b LVDS streams into one 5 Gbps CML serial signal. Backward compatibility to 1.25 and 2.5 Gbps speeds supports existing low and medium speed line cards while the 5 Gbps maximum rate enables the router to support high capacity line cards without chassis redesign.

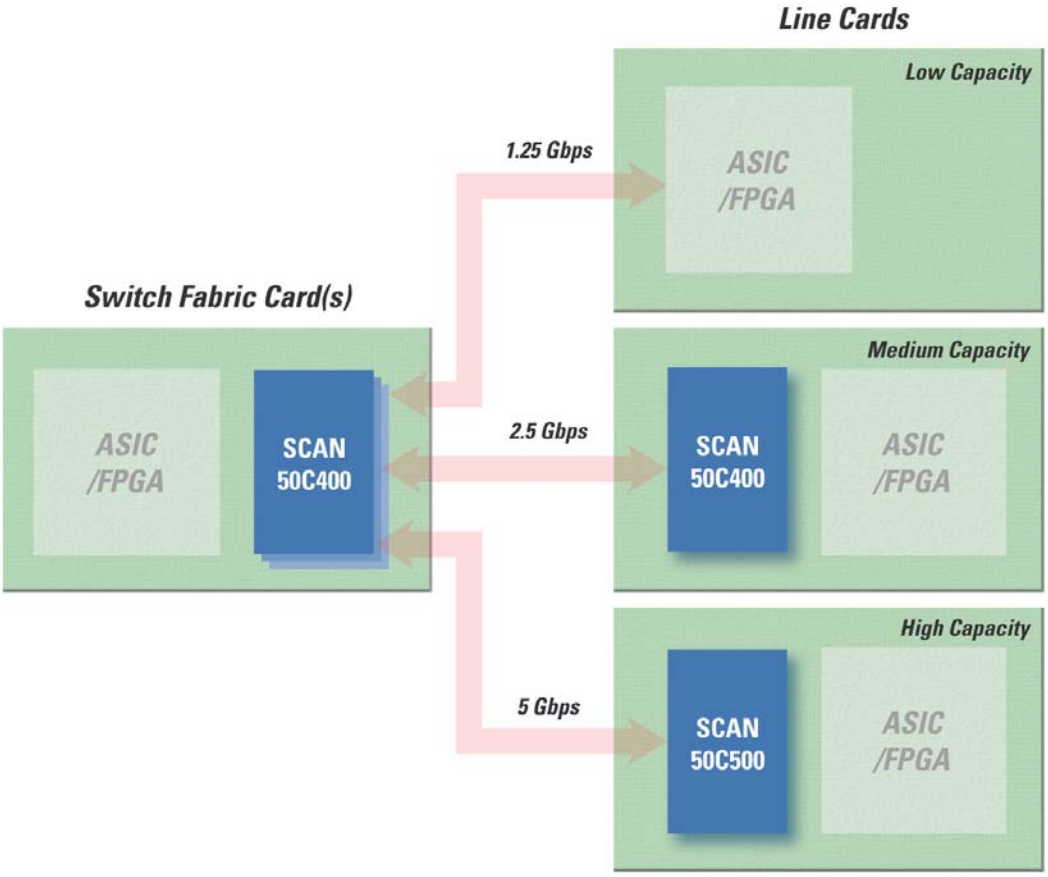


Figure 14. Bit interleaving SerDes router application example.

Comparison Overview

Each SerDes architecture has its own advantages making it well suited to certain applications.

Parallel clock SerDes are inexpensive and conveniently serialize wide buses. The major drawback is the use of multiple serial pairs, requiring more wiring and low pair-to-pair skew.

Embedded clock bit SerDes are well suited for applications needing a couple of extra bits and/or the lock to random data feature. They also have relaxed transmitter and reference clock requirements for systems with inexpensive clock sources. The lack of DC balance coding can be a disadvantage with AC coupling and when driving optical modules.

8b/10b SerDes work well with byte-oriented cell or packet data. The 8b/10b coding guarantees sufficient edge transition density in the serial stream as well as DC balance for driving AC-coupled interconnect and fiber optics. When using 8b/10b SerDes with bus

widths that are not byte multiples, extra design effort is required to pack the bus into bytes and run the SerDes link the increased speed.

Bit interleaving SerDes are required for many SONET/SDH applications and also datacom applications where it's imperative to get the most data through the fewest pairs. Such performance, however, comes at a higher price and tighter jitter requirements.

Technology	Advantages	Disadvantages
Parallel Clock SerDes	Serializes wide buses Low cost Automatic transmitter/receiver sync	More pairs/wires needed Tight pair-to-pair skew requirements
Embedded (Start-Stop) Bit SerDes	10- and 18- bit widths available Lock to random data capability Relaxed clocking requirements	No inherent DC balance Not well suited for AC-coupled or fiber applications
8b/10b SerDes	DC balance coding Works well in AC-coupled & fiber environments Widely available	Byte-oriented Tight clocking requirements Requires comma for sync
Bit Interleaving SerDes	Aggregates existing slower serial streams SONET/SDH-compliant versions	High speed design challenges Higher cost

Figure 15. Comparison overview of advantages/disadvantages of SerDes architectures.

Conclusion

Over the past ten years, several SerDes architectures have flourished to meet the diverging needs of a growing number of applications. Understanding the advantages and disadvantages of each allows the designer to fit the SerDes to the application versus the application to the SerDes in order to maximize performance and minimize system cost and complexity.

[1] A. X. Widmer and P. A. Franaszek, *A DC-Balanced, Partitioned-Block, 8B/10B Transmission Cod*e, IBM J. Res. Develop. Vol. 27, No. 5, 1983

[2] *LVDS Owner's Manual Design Guide*, National Semiconductor Corporation, 2001