

Slay Your System Dragons with JESD204B

by Ian Beavers, Applications Engineer,
Analog Devices, Inc.

The JESD204B serial data link interface was developed to support the growing bandwidth needs of higher speed converters. A third generation standard, it provides a higher maximum lane rate (up to 12.5 Gbps per channel) while supporting deterministic latency and harmonic frame clocking. Additionally, it now can easily move large quantities of data for processing by taking advantage of higher performance converters that are compatible and scalable with open market FPGA solutions.

FPGA providers have been talking about multigigabit serialization/deserialization (SERDES) interfaces for many years now. In the past, though, most analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) were not configured with these high speed serial interfaces. The FPGAs and the converters did not interface with any common standard that took advantage of the high SERDES bandwidth. JESD204B compliant converters can solve this problem, but this new capability introduces some questions.

What is 8b/10b encoding, and why is it needed on the JESD204B interface?

There can be no assurance of a dc balanced signal on a differential channel with random unencoded serial data, as there easily could be a larger number of either ones or zeros transmitted as opposed to the other. Random data that is sent across a serial link also has the potential to have long periods of inactivity or data that could be all ones or all zeros for a relatively long time.

When this happens, the dc balancing of an unencoded serial data stream becomes railed to one extreme or the other. At that point, when active data begins again, there is a strong potential for bit errors as the biasing of the lines is restarted. An additional long-term concern is electromigration, as a differential dc voltage is maintained on one side of the pair relative to the other. To counteract these issues, an 8b/10b encoding scheme is commonly used in differential serial data streams, including JESD204B.

In 8b/10b encoding, 10 bits are used to send the original 8 bits of information out of the source transmitter via a lookup table. This inherently accounts for a 25% inefficiency overhead ($10b/8b = 1.25$). In addition, the encoding allows

for at least three but no more than eight bit transitions per 10-bit symbol. This ensures that there are enough transitions for the receiver to recover an embedded clock, regardless of the dynamic activity of the underlying data.

The disparity between the number of binary zeros and ones in the serial stream is kept to within ± 1 using 8b/10b encoding, so the signal maintains a dc balance over time. The converse decoding of 10 bits to 8 bits must then be performed on the data stream at the receiver side to be able to recover the original data using the reverse lookup table. A more efficient 64b/66b encoding that operates on a similar principle, but with only a 3.125% overhead, is more advanced and has the potential to be used in future generations of JESD204.

The assigned JESD204B lanes of my converter do not route easily to my FPGA on my system board. There are crisscrossing pairs all over the place and it is generally susceptible to crosstalk. Is there a way to remap the assignment of the JESD204B lanes to make my layout easier?

Although converters may have JESD204B serial lanes defined by a number, letter, or other nomenclature to designate their particular relevance in the complete link, they are not required to be fixed. The specification allows for remapping of these assignments in the initial configuration data, as long as each lane and device has a unique identification. The link configuration data includes the device and lane identification numbers to identify its operation. With this information, a multiple lane transmitter could easily reassign any digital logical serial data to any physical output lane using a crossbar mux.

While it is an optional feature that the specifications allow, if an ADC vendor has a crossbar mux feature to reassign logical to physical output assignments, then the link I/O can be reconfigured in the best order for the easiest layout. The FPGA receiver can take the same initial configuration data and change the expected lane assignments to recover the data. With this ability, the routing of lanes from one device to the other can be made much easier and independent of the initial named assignment by the silicon vendor in the data sheet.

I am looking at potentially designing a converter into my system that uses a JESD204B multipoint link. How is it different from a single link?

The JESD204B specification makes provisions for what is known as a multipoint link interface. This is a communications link that connects three or more JESD204B devices. This link configuration can make sense over a single

link in some cases, depending upon how the converter is being used.

Take, for example, a dual ADC that uses JESD204B. In most cases, a dual ADC would have a single clock input to both converters. This would force simultaneous analog sampling at the same frequency. But for some unique applications, such a device could also use two separate input clocks, where each clock could drive its respective ADC independently. This would allow for a sampling phase difference between the two ADCs, or even for each ADC to be sampled with a noncoherent frequency with respect to the other. In the latter case, a single JESD204B link with data from both converters would not operate correctly without a complex back-end FIFO scheme.

A solution to this problem could be to have the dual converter use a multipoint link JESD204B interface, where each converter channel uses its own serial link output. Noncoherent clocks then could be used on each ADC, and each serial link output could easily route independently to a separate FPGA or ASIC. A multipoint link configuration can also be used when sending multiple streams of data from a single FPGA to several DACs. Device clock distribution skew can be more challenging to minimize within a multipoint configuration as the number of devices within the link grows.

What exactly is deterministic latency within JESD204B? Is this the same as the total latency of my converter?

The total latency of an ADC is the time it takes an analog sample to be clocked in, processed, and output digitally from the device. Similarly, the total latency of a DAC is the time from when the digital sample data is clocked into the part

until that corresponding sample is clocked out of the analog output. Typically, these are both measured in sample clock periods of resolution, as they are frequency dependent. This is fundamentally not the same definition as the deterministic latency described by a JESD204B link implementation.

Deterministic latency across the JESD204B link is defined by the time it takes data to propagate from the parallel framed data input at the transmitter (ADC or source FPGA) to the parallel deframed data output at the receiver (DAC or receiver FPGA). This time is typically measured in either frame clock periods of resolution or device clocks (Figure 1). The definition excludes the analog front-end core of an ADC and the back-end analog core of a DAC. Not only are two devices a function in this latency computation, but so is the serial data signal routing interfacing the two. This means that the deterministic latency could be larger or smaller within a multiconverter system or multipoint link, depending upon the length of the JESD204B lane routing. Buffer delays on the receiver can help account for latency differences due to routing.

How are tail bits used in JESD204B, and what is their purpose?

The JESD204B link allows for more information space to be allotted than may actually be needed to send the converter data and control bits. If data for a particular converter or configuration does not fill up the entire space, then this “padding” is filled with what are defined as tail bits. Take, for example, a case where a space of $N' = 16$ is more than the parceled 13 bits of real data ($N = 13 + CS = 0$). Three tail bits would be used to fill the unused data space (Figure 2).

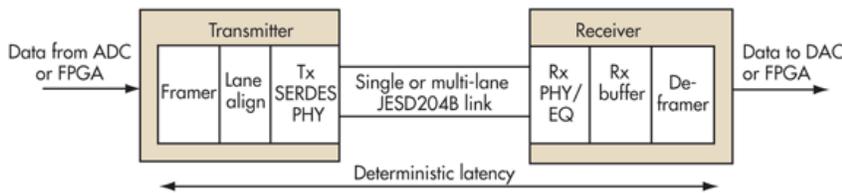


Figure 1. A Conceptual Example of JESD204B Deterministic Latency Between Framer and Deframer on Two Linked Devices. The Latency Is a Function of Three Items: the Transmitter, the Receiver, and the Interface Propagation Time Between the Two

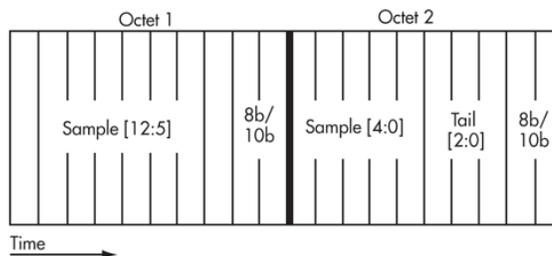


Figure 2. Three Tail Bits Can Be Used to Pad and Fill a Second Octet for $N' = 16$ When a Converter Only Uses 13 Bits of Sample Data

Tail bits are informationless dummy bits and are only used to completely pad the unused space. Since tail bits have the potential to cause unwanted spurious noise if they are all assigned a recurring static value, they also can be optionally represented as a pseudorandom sequence. Both the transmitter and receiver must understand that these bits are informationless based on the link configuration. The receiver can therefore simply discard them from the stream of relevant data.

My link patterns work fine, but I am not getting converter data transmitted in a normal operating mode. In past generations of converters, low voltage differential signaling (LVDS) and parallel interfaces allowed easy probing/debugging of a least significant bit (LSB) or most significant bit (MSB) pin of a DAC or ADC to see if the functional converter operation was taking place. How can I probe an MSB or LSB when using the JESD204B interface?

This is one of the few drawbacks to the JESD204B interface. It is not easy to electrically probe an LSB or MSB I/O to see if there is correct activity to and from the converter. This is because the sample data is serialized per channel, so a particular weighted bit cannot easily be probed electrically. However, a few options can be used to debug a system issue when you quickly want to know what, if any, valid data is being sent or received from your converter.

Some oscilloscope vendors provide real-time data processing to serially decode 8b/10b data and display an unencoded stream on the oscilloscope screen. Unscrambled data can be probed in this fashion to determine what activity is taking place on the link.

FPGA vendors offer an internal probing software tool that gives system designers a method to observe the I/O data sent and received from within the FPGA by connecting it via a USB dongle to a computer. Also, some ASICs and converters offer an internal serial loop-back self-test mode that can be used to help decipher data issues on the link.

How do I calculate the lane rate for my converter, given that I know the other parameters of the link?

System designers using JESD204B can easily compute the number of lanes or lane rate for their link given that they know the other key criteria of their converter, ASIC, or FPGA. There is a mathematical relationship for all of the basic link parameters below such that one unknown variable

can be computed and solved. Based on the result, system designers can choose other parameters to change the link operation within the confines of the converter or FPGA architecture:

$$\text{Lane rate} = (M \times N' \times [10/8] \times Fs) / L$$

where:

M is the number of converters on the link.

N' is the number of informational bits sent in a sample (including sample resolution, control and tail bits).

F_s is the device or sample clock.

L is the lane count.

Lane rate is the bit rate for a single lane.

10/8 is the link overhead due to 8b/10b encoding.

For example, consider a dual ADC with *N'* = 16, *F_s* = 235 MHz, using two lanes. What is the lane rate?

$$\text{Lane rate} = [2 \times 16 \times 1.25 \times 235 \text{ MHz}] / 2$$

$$\text{Lane rate} = 4700 \text{ Mbps or } 4.7 \text{ Gbps}$$

What is an application layer, and what does it do?

An application layer is a method provided for in JESD204B that allows sample data to be mapped outside the normal specification. This can be useful for certain converter modes that need to pass data samples in sizes that are relatively different from the *N'* of the link.

An otherwise inefficient arrangement on the link can be made more efficient with a lower lane count or lower lane speed by using an application layer. Both the transmitter and receiver need to be configured to understand a specific application layer, as it can be custom or uniquely designed by a particular converter mode. Figure 3 shows an example where five samples are partitioned into a space typically occupied by only four.

When using the equation from the previous question for application layer calculations, the effective *N'*, instead of the actual *N'*, needs to be used. For example, in the application layer case shown below, although the actual JESD204B sample *N'* is 16, the effective *N'* for ADC samples can be figured since 64 bits are used to send five samples. Therefore, *N_{eff}* = 64/5 = 12.8. With all other variables held equal, the lane rate could then be run 20% slower:

$$N_{eff} / N' = 12.8 / 16 = 0.8.$$

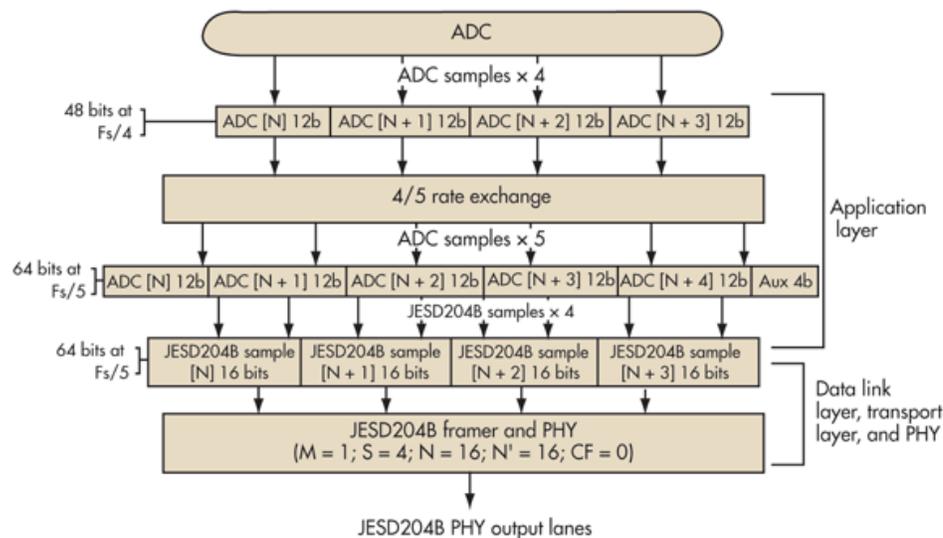


Figure 3. An ADC Application Layer Can ReMap Five 12-Bit ADC Samples into the Space Used by Four JESD204B $N' = 16$ Samples. Four Bits of Extra Auxiliary Information Can Be Made Available for Additional Use

What's next?

As JESD204B continues to proliferate within the data converter market, intellectual property (IP) capabilities on FPGA platforms should help speed its adoption. Future discussions on the subject should only grow, albeit with additional complexity, as more engineers become involved and start designing new systems.

ABOUT THE AUTHOR

Ian Beavers is an applications engineer for the High Speed Converters team at Analog Devices, Inc., Greensboro, N.C. He has worked for the company since 1999. He has more than 15 years of experience in the semiconductor industry. He earned a bachelor's degree in electrical engineering from North Carolina State University and an MBA from the University of North Carolina at Greensboro. He can be reached at Ian.Beavers@analog.com.

REFERENCES

JESD204B. www.analog.com/jesd204.

RESOURCES

Share this article on

