

# Maximizing Performance and Integration in Applications Requiring Isolated SPI

by Mark Cantrell and Bikiran Goswami, Analog Devices, Inc.

## Real-World SPI Implementation

SPI is a very useful and flexible standard, but its flexibility stems from its simplicity. Four channels of unidirectional moderate speed isolation will handle SPI to clock rates of a couple of MHz. What a designer gives up are features like interrupt service support and direct communication channels like a reset function, or alarms from non-SPI enabled functions. The result is that practical interfaces that use SPI frequently have extra GPIO lines running in parallel to handle these functions. When the SPI is isolated, all of these lines need isolation. In many cases the extra communication does not require high speed, so the designer goes off to the approved parts list for a couple of optocouplers to drop next to the high speed digital isolator required for SPI. But optocouplers are not as simple to design with as a digital isolator, especially simple low speed optocouplers. A crash course in how optocoupler CTR (current transfer ratio) changes with time and temperature and how it affects speed in the application may be required. If the extra communications are in multiple directions, then multiple packages will be required, since optocoupler channels must transfer data in the same direction if it is a multichannel device. Size, cost, and time invested increase rapidly as the design is worked out.

## Isolated SPI Using *iCoupler*<sup>®</sup>-Based Digital Isolators

An SPI bus is easily isolated using four channels of high speed digital or optical isolation. Quad digital isolators enable isolation in a compact form factor while typically supporting 5 MHz to 10 MHz SPI clock rates. Whether based on integrated microtransformers, capacitors, or optocouplers, signals need to be transferred across the isolation barrier. Among current isolation technologies, two primary encoding schemes exist to couple the input signals: edge encoding and level encoding. The level encoding schemes allow lower propagation delay isolators, and generally consume higher idle power and have poor timing performance (jitter and PWD). The edge encoded schemes on the other hand have lower power consumption and better timing performance. However, many edge-based implementations have longer propagation delays, which can limit clock speeds in SPI buses. We will explore techniques to eliminate this limitation and allow the isolation of very high performance SPI buses.

In edge encoded schemes, the input digital transitions are encoded across the isolation barrier. The digital input is then reproduced at the output side by decoding these transitions. Example waveforms (Type A) in Figure 1 show how the input edges can be encoded differentially. Rising and falling input transitions are encoded as narrow positive and negative pulses, respectively.

This signal transmission works fine as long as the digital input is switching, but if input to output dc level mismatches occur, they can persist unless they are corrected. This is most often apparent during power-up when the data channel is idle; nothing is communicating the dc state of the input pins across the isolation barrier and the startup state might not match the input data until an edge is transmitted. To solve this, edge encoded schemes employ some form of “refresh” circuitry to ensure dc correctness at the outputs. Refreshing operates by periodically transmitting the last transition across the barrier after a preset duration ( $T_{\text{REFRESH}}$ ) of input inactivity. Figure 1 also shows refreshing in the example edge encoding scheme (Type B waveform).

In the waveform, even though the input signal hasn’t transitioned between the rising and falling edges, multiple encoded rising transitions are transmitted across the barrier to refresh the receiver state and ensure it at the high state. It’s easy to see that, besides ensuring correct output states on power-on, the refresh circuits also help correct the output state if it gets corrupted and goes out of sync from the input state. This scheme consumes idle power since encoded data is constantly transmitted.

Since refreshing is required in edge encoded scheme, it’s useful to understand the timing impact inherent in this scheme. From the waveforms in Figure 1, we see that both the input transitions and refresh signals are transmitted over the same isolation channel, but are separated in time. As the input signal is completely asynchronous to the internal refresh generation circuitry; it’s possible for an input transition to occur at the same time a refresh for the previous transition is being transmitted. This could result in a timing hazard at the receiver causing a logic error. To avoid this timing collision error, the refresh circuitry requires adding some “look ahead” delay in the signal path. The delay ensures a minimum separation between encoded signals across the barrier, therefore allowing the receiver to unambiguously decode any sequence of coded transmissions. The drawback is that this delay

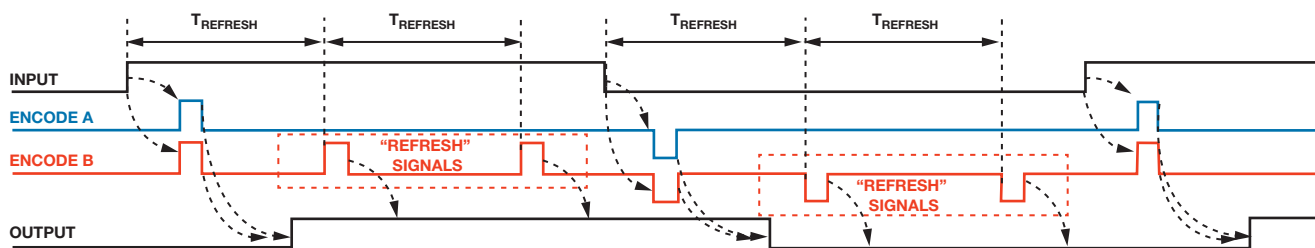


Figure 1. Example of encoded waveforms across isolation barrier. Type A and Type B are encoding waveforms with and without refresh signals, respectively.

can significantly increase the propagation delay of the isolator and therefore limits its ability to meet SPI timing constraints as speed increases.

Fortunately, this limitation can be addressed. If the input transitions and the refresh states are transmitted on different isolation elements and later merged at the output, there isn't a need for any look ahead delay and its accompanying timing penalty. Extending this idea to multichannel isolators, the refresh states for all the channels could be time multiplexed over a single isolation channel, and later demultiplexed and merged with their respective outputs. Basically, the input states are sampled, packetized, and serially transferred across the isolation barrier. The receiver keeps track of the input dc state and then based on how long the input has remained inactive, arbitrates between using the dc state or the last input transition to update the output. With just one extra digital isolator channel to carry all the refresh states, we can free all the input channels from look ahead delays and let them just carry the input switching information, thus significantly improving propagation delay. This idea is illustrated in Figure 2.

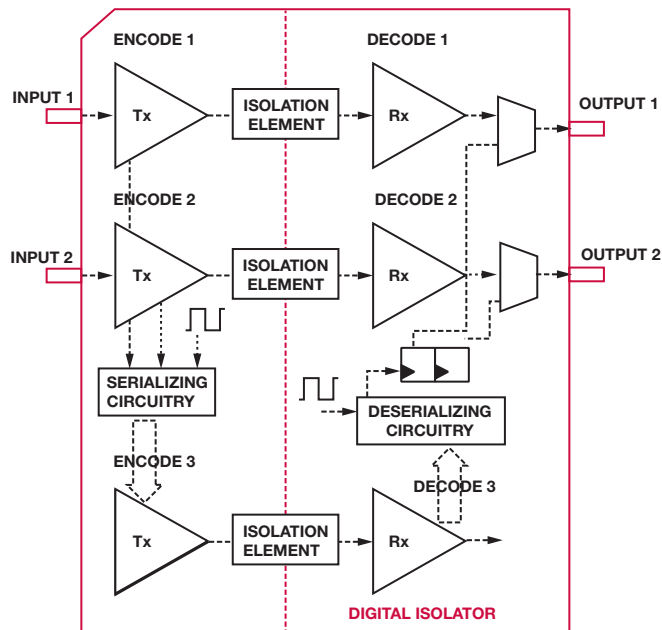


Figure 2. Block diagram with dedicated 3rd isolation channel carrying the time multiplexed refreshes of Input 1 and 2.

The Analog Devices SPIsulator® series of high speed digital isolators use this scheme to achieve very low propagation delay on the SPI channels, thereby enabling high bandwidth isolated SPI bus implementations, up to 17 MHz. The extra isolator channel carries refresh information, and is additionally used to transfer a variety of other low throughput signals, by time multiplexing the data into packets transmitted back and forth across the isolation. This enables additional communication besides the SPI serial data bits, implementing a versatile yet highly integrated isolated SPI bus.

## SPI with Aux Data Channels

Some versions of the SPIsulator product family bundle three 250 kbps auxiliary (aux) digital channels on the multiplexed refresh channel. Three different product variants allow multiple channel direction configurations for those aux channels. Even though the aux channels are asynchronous, they are sampled and packetized prior to transmission over a single isolation channel. Based on when the channel inputs switch with respect to the internal sampling clock, the propagation delay of these aux channels can vary up to a maximum of 2.6  $\mu$ s. Sampling and serializing the asynchronous aux signals also synchronizes them to the internal sampling clock. This can result in timing issues if accurate timing between the slow signals is important, especially for accuracies within a time window of 1.2  $\mu$ s or less. Fortunately, very few auxiliary signals surrounding typical SPI buses require such accurate timing relationships between one another. More importantly, this low speed data system is carefully designed to preserve edge order (of multiple asynchronous channels) as long as edges are separated by at least a minimum skew ( $V_{IXSKEW}$ ). In other words, if one edge is leading another at the input, the order is not reversed by the isolator.

## Common SPI Interface Example

In Figure 3 we can see a typical SPI application requiring a 1 MHz SCLK, 4-wire SPI, and three additional signals for interrupt, power good, and reset. The low speed channels may require only 40  $\mu$ s prop delay. These parameters were chosen to allow all component options to be examined. The timing is well within the capabilities of all of the major isolator types so that we can compare the effect of integration on the solution size and cost independent of performance. Here are some implementation options:

- 1) Use all optocouplers
- 2) Use a digital isolator for the SPI and optocouplers for the slow signal channels
- 3) Use a fully integrated digital isolator like the ADI SPIsulator

Schematically, the solutions do not look very different. However, if we look at Figure 4, we can see how the ICs might be laid out on a PCB. The red box approximates the area required for the isolation components, as well as allowing space for required passive components.

Starting with the mixed speed optocoupler solution as the baseline, many designers will look at combining standard digital isolators with very inexpensive optocouplers, thinking that is the most cost effective way to implement low speed signaling. A large part of the difference in area between solutions occurs because a digital isolator allows mixed channel direction within a single package, avoiding a lot of packaging overhead. Mixed technology solutions can be cost effective, but with extra design time penalties, and stability issues. Inexpensive optocouplers are slow and because of their simplicity, they require some thought and research to build a successful design. Care must be taken to be sure that they will be stable over time and temperature, while minimizing their power consumption.

The reduction in area moving from an optocoupler solution to a fully integrated solution like the ADI ADuM3152 SPIsulator is almost 75%.

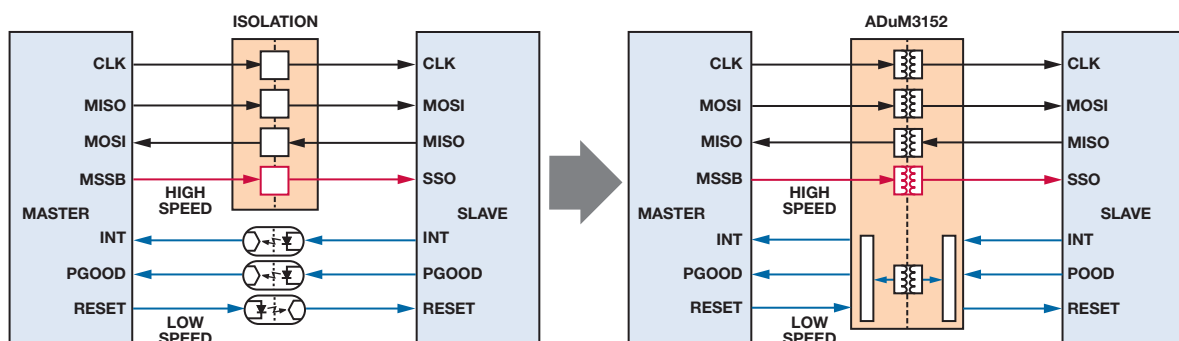


Figure 3. Typical SPI application with supplemental functions implemented with multiple technologies.

The integrated solution available in the SPIsolator has built-in protection from channel to channel speed differences and pulse order reversal, with no design overhead, and no extra board space. The low speed channels available in the integrated solution also support data rates that are significantly higher than single transistor optocouplers. The cost for the integrated solution is well under half that of the optocoupler solution and provides the low speed channels at lower cost per channel than discrete optocouplers. Considering SPI performance, the ADuM3152 digital isolation solution can support SPI with up to 17 MHz clock rates because of the very short propagation delays while optocoupler “fast” channels can work up to 3 MHz at reasonable cost.

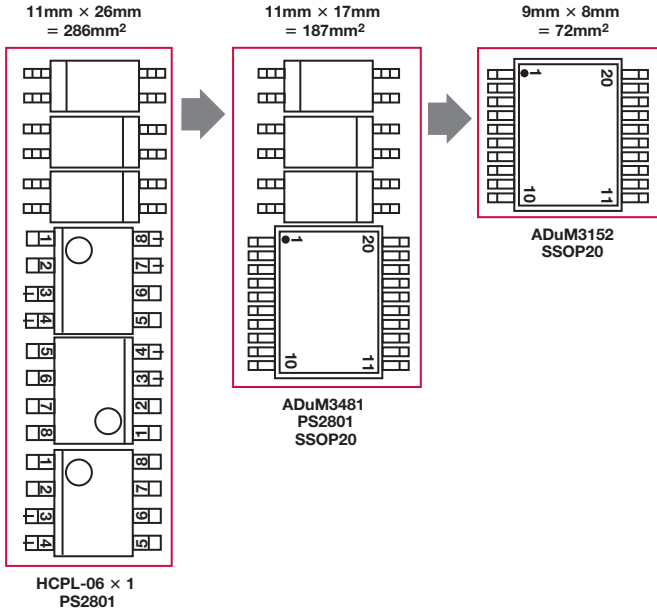


Figure 4. Typical SPI application with supplemental functions implemented with optocouplers.

### Demultiplexing SSB to Control up to 4 Slaves

In the ADuM3154 product variant, a 250 kbps, low speed, 2-channel address bus is used to control the isolated slave select address lines (SSx), allowing the target slave devices to be changed in as little as 2.6 μs. Figure 6 shows implementations using general-purpose isolators and the ADuM3154. Figure 5 shows how the SSx transition as the address bits (SSAx) are changed.

The ADuM3154 uses a 2-bit address bus to route the master slave select (MSS) to one of four slaves. These address bus bits are slow speed signals that are once again bundled along with refresh states of the regular four high

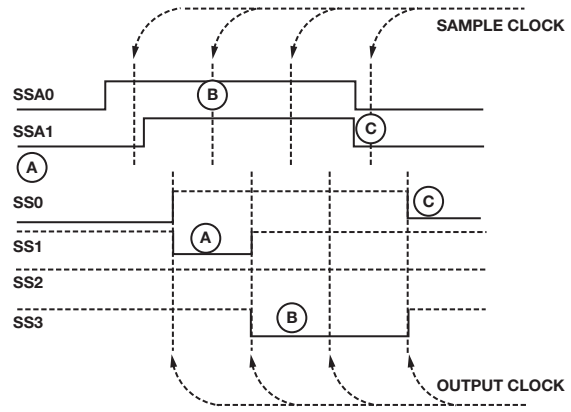


Figure 5. Slave select address bits SSAx select one of four slaves SSx.

speed SPI channels. Like the refresh states, the address bits are all sampled, packetized and serially transmitted across the isolation barrier. On the slave side, the packets are deserialized and the address bits are used to demux the MSS. Depending on when the address bus toggles with respect to the internal sampling clock, the demultiplexer routes the MSS signals to the desired slave within 2.6 μs. The address bits are part of a bus and must be in sync with each other. Special care is taken during sampling and serializing to ensure that these address bits are kept in sync at the output side, especially during transition from one selected slave to another.

### Multiple Slave SPI Interface Example

A common SPI design requirement is talking to several slave devices sharing the same SPI bus. This can be done in a couple of ways. If the data will be sampled from all secondary side devices simultaneously and all data is transferred every frame, then the simplest thing to do is daisy-chain the parts together and shift the entire contents of the chain out serially through a single isolated port. But when the sequence of data acquisition is not fixed, then each SPI slave must be addressed individually. This offers special challenges for isolated interfaces.

If each slave must be addressed separately, then each device must have an individual slave select line. In many cases the slave select not only selects a specific target for SPI communication, but also initiates conversions in an ADC for example, so this line must also have highly accurate timing. In many implementations, this requires an extra isolation channel of comparable speed to the clock channel to preserve timing. A 4-channel isolated SPI design is shown in Figure 6 where the standard four channels of high speed isolation are augmented with three extra channels of high speed isolation.

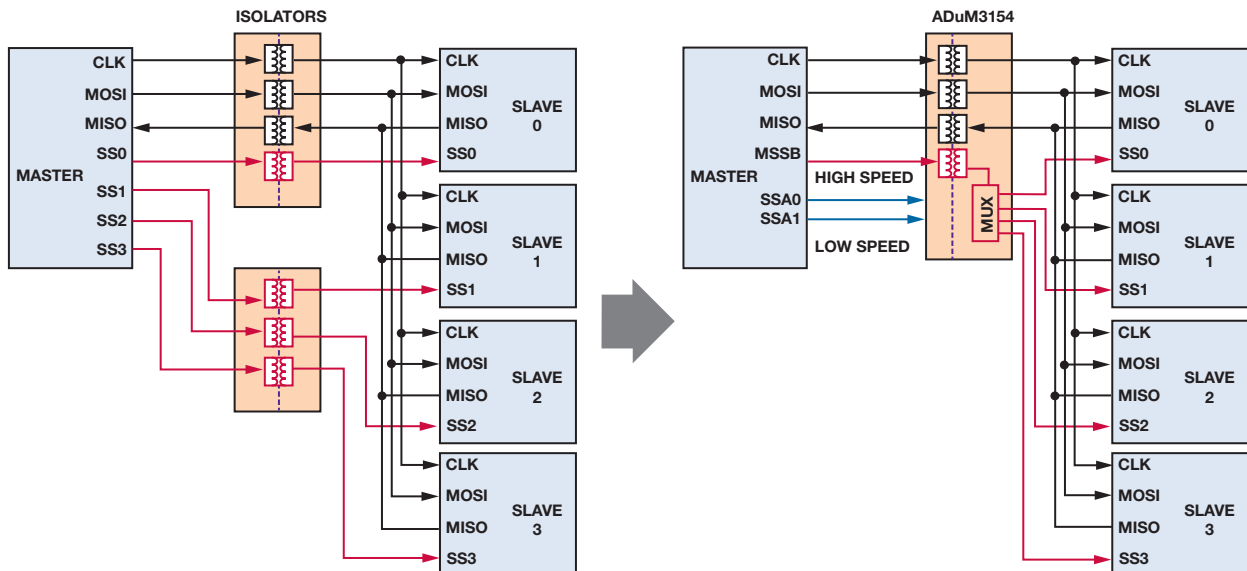


Figure 6. Typical multiple slave SPI application.

Another approach to slave selection is shown in the right hand side of Figure 6. A multiplexer is used on the secondary side and lower speed isolated select lines can be used to select the destination. Care must be taken to switch the multiplexer control lines so that they will not jump to an incorrect intermediate state if the timing is a little off. This scheme is implemented with the low speed channels available in the SPIsulator devices, and since the application is well defined, safeguards for indeterminate states can be built in to prevent small timing errors from creating transient output states.

In the final analysis, when performance can be supported by any technology, the best design choice comes down to ease of implementation, size, and cost. Three possible implementations are shown in Figure 7. From left to right is a simple optocoupler solution using seven channels of isolation, next is the same scheme using digital isolators, and finally the integrated multiplexer function of the SPIsulator. The optocoupler solution is the largest while the digital isolator is 72% as large and the SPIsulator takes up only 36% as much PCB real estate. The cost for the three solutions scales similarly, with the SPIsulator approach significantly cheaper than the alternatives.

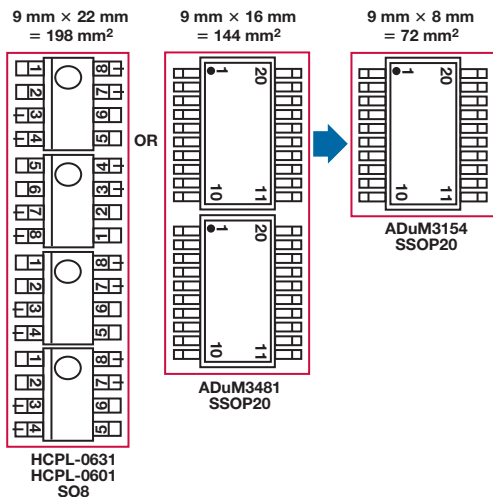


Figure 7. Typical SPI application with supplemental functions implemented with optocouplers.

## Conclusion

In designing a high performance isolated SPI solution, the SPIsulator product family uses a multiplexed control channel to support high speed SPI with very low propagation delay, and maximizes integration of auxiliary functions. This combination of SPI channels with supplemental functions allows convenient design of highly integrated isolated SPI buses while reducing design time, cost, and board space.

For more information on any of the above mentioned products visit [www.analog.com/DAC](http://www.analog.com/DAC).

## About the Authors

Mark Cantrell is a staff applications engineer for the *iCoupler* Digital Isolator Group at Analog Devices, Inc. (ADI). His area of expertise is *iCoupler* digital isolation products, including *isoPower* isolated power supply devices and communications bus devices such as I<sup>2</sup>C and USB isolators. He is also responsible for agency safety certifications for all *iCoupler* digital isolator products. Mark received his MS in physics from Indiana University. He can be reached via email [mark.cantrell@analog.com](mailto:mark.cantrell@analog.com).

Bikiran Goswami is a mixed-signal integrated circuit design engineer for the *iCoupler* Digital Isolator Group at Analog Devices, Inc. (ADI). His area of expertise is in designing *iCoupler* digital isolation products aimed at the standard data isolation devices and communication buses. Bikiran received his MS in electrical engineering from Stanford University. He can be reached via email [bikiran.goswami@analog.com](mailto:bikiran.goswami@analog.com).

## Online Support Community

Engage with the Analog Devices technology experts in our online support community. Ask your tough design questions, browse FAQs, or join a conversation.

[ez.analog.com](http://ez.analog.com)



**Analog Devices, Inc.**  
**Worldwide Headquarters**  
 Analog Devices, Inc.  
 One Technology Way  
 P.O. Box 9106  
 Norwood, MA 02062-9106  
 U.S.A.  
 Tel: 781.329.4700  
 (800.262.5643,  
 U.S.A. only)  
 Fax: 781.461.3113

**Analog Devices, Inc.**  
**Europe Headquarters**  
 Analog Devices, Inc.  
 Wilhelm-Wagenfeld-Str. 6  
 80807 Munich  
 Germany  
 Tel: 49.89.76903.0  
 Fax: 49.89.76903.157

**Analog Devices, Inc.**  
**Japan Headquarters**  
 Analog Devices, KK  
 New Pier Takeshiba  
 South Tower Building  
 1-16-1 Kaigan, Minato-ku,  
 Tokyo, 105-6891  
 Japan  
 Tel: 813.5402.8200  
 Fax: 813.5402.1064

**Analog Devices, Inc.**  
**Asia Pacific Headquarters**  
 Analog Devices  
 5F, Sandhill Plaza  
 2290 Zuchongzhi Road  
 Zhangjiang Hi-Tech Park  
 Pudong New District  
 Shanghai, China 201203  
 Tel: 86.21.2320.8000  
 Fax: 86.21.2320.8222