Streamlined Debugging of SPI Using Mixed Signal Oscilloscope

Introduction

Serial Peripheral Interface (SPI) is a synchronous serial communication interface that is used in many embedded systems. It's far more efficient than a parallel I/O bus, so it makes a lot of sense that it is used in FPGA (Field Programmable Gate Array) designs. You see SPI used to communicate with a variety of peripherals: any sort of touch, temperature, or ADC sensor; camera lenses; communications (ethernet, USB, etc.); or control devices to name a few. With such a wide variety of applications, SPIs require troubleshooting for different functions. If you are designing a system that talks to a central control chip and some sort of peripheral like the ones mentioned above and the SPI bus isn't working properly, you will need to debug it before you can continue working on the system. By using a mixed-signal oscilloscope with four channels, you can directly look at and analyze each of the hardware pins connecting one chip to another and diagnose the specific issue you are running into.

Setting Up the Hardware

To debug a SPI, you'll need a system with a central control chip. In the example we will be referring to throughout this whitepaper, we are using an FPGA development board (the type doesn't necessarily matter, just that it has a control chip and some mounted peripherals that you can access with probes) with a Pmod TPH daisy-chained to a Pmod AD1 acting as the sensor in the system (see below). We are also using the Analog Discovery Pro ADP3450 as a test instrument in this example.

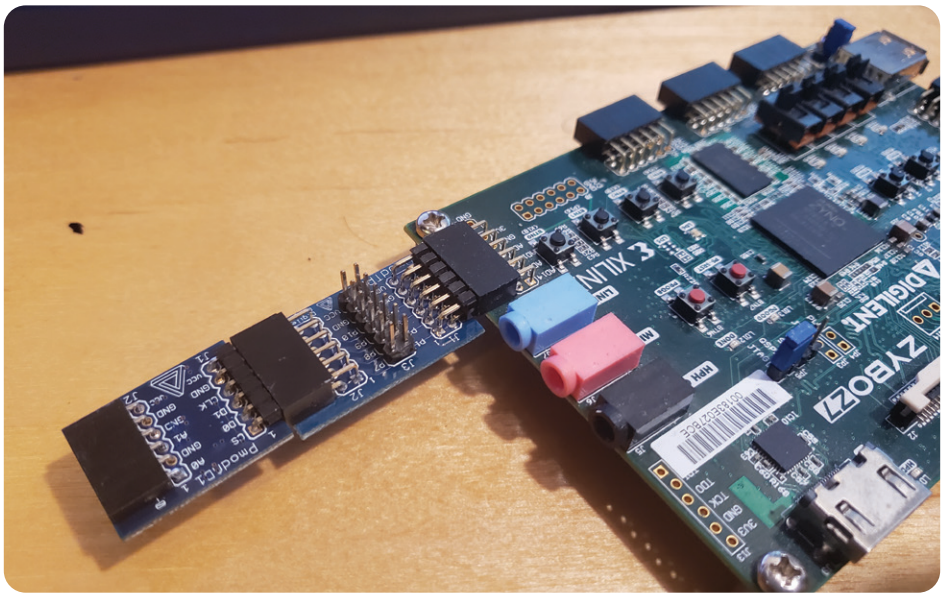

Figure 1: A SPI device connected to a development board

First, you will need to connect BNC probes from your test instrument to your test points on the circuit that you are trying to debug, which are where your physical wires are exposed. Typically, there is one test point per signal – and you might need to use additional devices, such as chip clips or a breakout like the test point headers that can be seen in the image above. Once you have done this, it should look more or less like the image below.

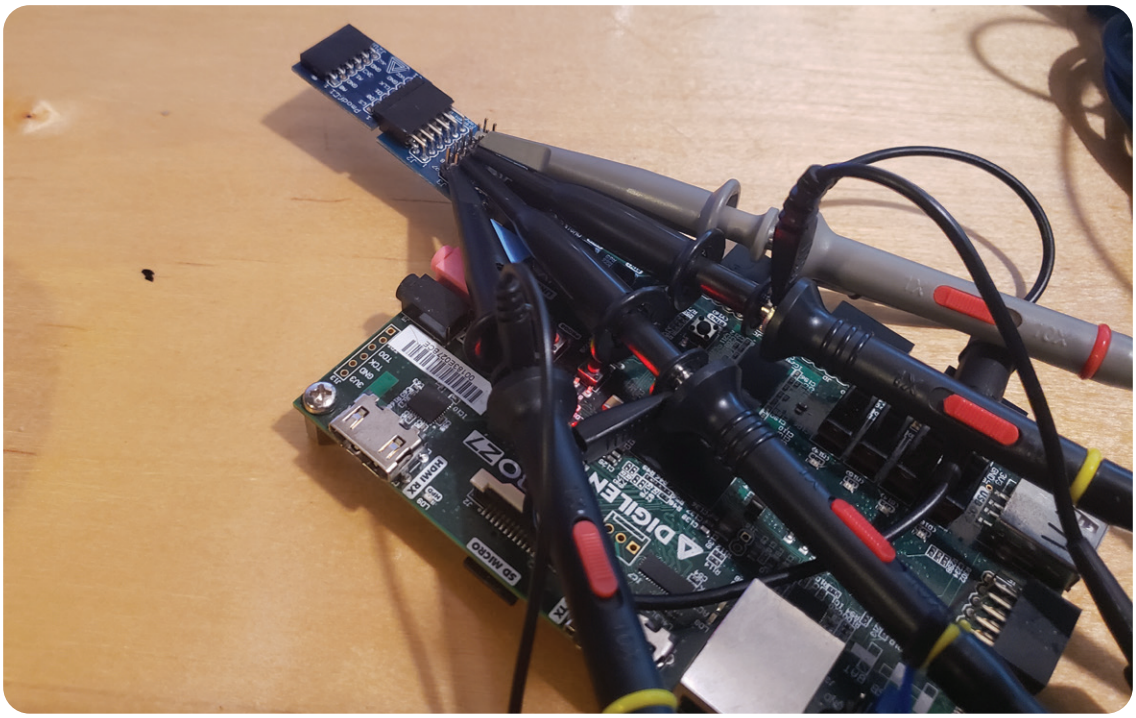

Figure 2: BNC Probes connecting Test Instrument and Circuit

Scope Configuration and Software Setup

Channel Configurations

Especially when looking at four channels that may be following similar lines through a plot, it's important to make sure you can see each of the signals of interest. In Figure 3 (below), you can see that the offsets and ranges of each signal have been chosen to make sure that the signals do not overlap and are visually distinct. In this case, a two-division horizontal band was given to each, effectively splitting the plot pane into four rows, one for each signal. The time base and position were also chosen to ensure that each clock edge could clearly be seen, as well as the chip select edges at the beginning and end of the frame. The time position depends on the trigger source, which will be discussed later.

Selecting a BNC Attenuation

Next, select a BNC attenuation. The BNC probes that you use with your oscilloscope have an attenuation associated with them. For the ADP3450, these can either be set to 1X or 10X (with a corresponding dropdown to select in WaveForms). The primary difference between the two comes down to the amount of additional load you are placing on your circuit, which can influence the quality of the signal received. Which setting you choose will depend on the details of the system being debugged, and how sensitive it is to a load, but it likely doesn't matter much.

Triggering

Choosing a signal to trigger on, and even what the trigger condition is, entirely depends on what it is that the engineer is trying to look at/accomplish. To ensure that any sort of data is being transmitted between the host and the peripheral, trigger off chip-select going low. From here, check to see if there's any data going down MOSI or MISO. You may also be trigger off the clock transitions in some way to check that it is set to the right frequency.

Once you have confirmed that data is moving between host and peripheral, check that the right data is being sent. For that, we are using a value trigger on MOSI or MISO that fires based on some command pattern in our example. The SPI protocol used by the Pmod AD1 differs a little bit from other devices, in that there are two MISO data lines, and no MOSI. Each 16-bit frame sent over each of these lines represents the voltage value last read by the Pmod's analog-to-digital converters. The accuracy of the received data can be checked by first ensuring that a grounded ADC input produces near-zero readings. Using complex triggering methods, including scripts and logic analyzer digital inputs, you could, for example, check that the first four bits returned by each of the ADCs are in fact always four leading zeroes, as specified by the device's datasheet. For other SPI devices with register-based interfaces, you may also want to trigger on only reads or writes to specific registers – say a device requires you to set a specific bit in a specific register before it performs some function. You can trigger on only that write, and use other oscilloscope or logic analyzer channels to capture the chip's response to that stimulus.

At times it is important to see if the SPI bus signals meet the right setup and hold

times. The former is the amount of time that a data signal must be stable prior to the edge of the clock transition, while the latter is how long the signal needs to stay stable after that edge. Using the clock transition as the trigger centers the SPI signals in the scope window and makes using Cursors to measure setup & hold times simple and easy.

When validating analog properties like this, it's often a good idea to have the scope trigger on the falling edge of the chip select signal – of the four signals, the one that most infrequently transitions. You can still use horizontal offsets to scroll around and look at pieces of the signal which may not appear on the same screen at the same zoom level as the falling edge of the chip select. You will get the most consistent results from capture to capture this way – the 7th bit of a frame will typically fall around the same time offset, rather than requiring you to hunt for it in a capture that triggered on a clock edge. Once you have validated analog parameters and want to start looking at multiple-frame sequences, in the same way that triggering on the chip select is useful for observing sequences of bits in a single frame, the logic analyzer is beneficial for viewing sequences of frames. For example, viewing the data returned after a read command - potentially a multiple-word automatically incrementing read command at that - can come in a separate frame.

Interpreting Results

SPI Clock Polarity

The first step in interpreting results is visually checking the signals. Depending on the mode of operation, a SPI clock signal would typically idle high or low while the chip select line is not being pulled low, and data signals transition on either the rising or falling edge of the clock. Each of these actions can be seen clearly by examining your first capture.

Identifying Signals

- The chip select idles high and transitions infrequently compared to the others
- The clock transitions most frequently
- The data lines transition separates from the active clock edges with patterns wholly dependent on the data being transmitted

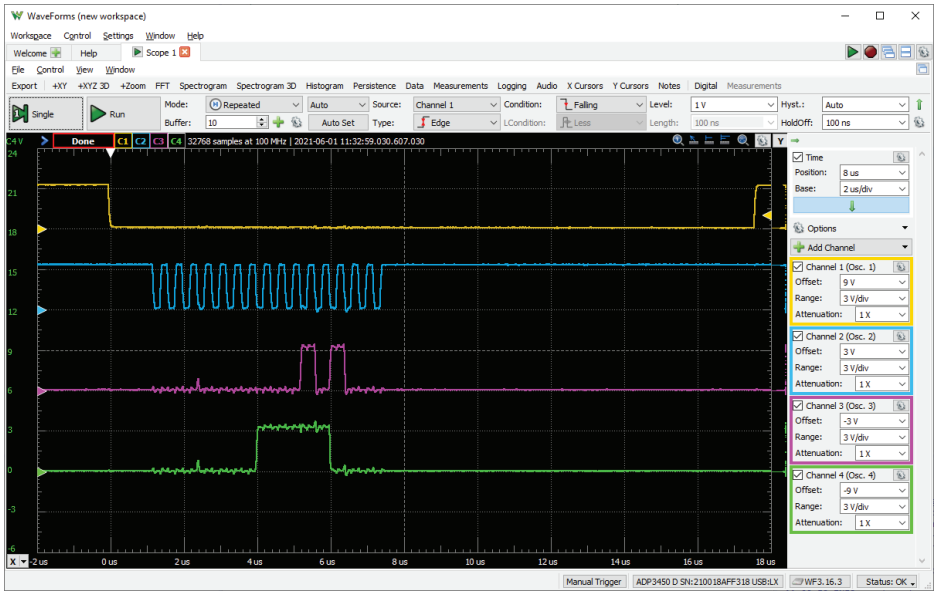

Figure 3: SPI capture with chip select (channel 1, yellow), serial clock (channel 2, blue), and two data lines (channels 3 and 4, purple and green)

SPI Clock Frequency

SPI devices typically specify a range of frequencies where they can operate – sometimes this could be just for the hardware running the serial interface, or sometimes it could be the clock from which the entire device runs. Regardless, it is commonly specified, and the clock provided must meet this spec for the device to perform as advertised. Most scopes have built-in measurement tools for analyzing various properties of captured signals. One of the more common tools can measure how long a pulse takes to occur, from rising edge to rising edge, or even just to outright measure the frequency of a signal with such pulses. Automatic measurements, visual estimation (using on-screen divisions), and cursors can all be used to accomplish this.

Figure 4: Measuring a 2.5 MHz SPI clock's frequency with a pulse measurement tool.

Additional Checks

The screenshot above shows a transfer of 16 bits representing 0x1010, with the most-significant bit (MSB) first. You can also observe that SPI Mode 3 is implemented, since the data signal transitions on the falling edge of a clock that idles high. You can't tell the correct bit order or the SPI mode that must be implemented from the scope, so check the documentation for the device you are working with.

Digilent and the Analog Discovery Pro ADP3450

With four high-resolution channels, the new Digilent ADP3450 serves as a capable companion to helping in the debugging of an SPI bus. With four BNC probes, the device's channels match up perfectly with SPI's 4 pins and you will be able to view all signals simultaneously using the WaveForms software. Within WaveForms' Script Editor, you can convert analog data into the frames of digital data needed to design your system.

For more information about debugging using FPGA-based development tools, contact Digilent at **sales@digilent.com**.

Since 2000, Digilent (a wholly owned subsidiary of National Instruments) has provided embedded engineers, researchers, scientists, and students with cost-optimized products, tools, and application information for innovative, FPGA and SoC based hardware-software systems. Our customizable and flexible solutions will accelerate development time for even the most experienced professionals, while maintaining low barrier to entry for advancing engineers, students, and the perpetually curious.

From our competitive pricing to the portability of our products and comprehensive documentation, we value delivering accessibility and lowering barriers to progress for our customers.

We specialize primarily in Xilinx-based FPGA/SoC development boards/kits and portable USB test and measurement devices, all designed to be owned by and used from an engineer's or student's desk. We also offer a variety of expansion modules (Pmods and Zmods) to create flexible I/O options for our other products.