

# Using SPI protocol at 100 MHz

## Understanding the constraints of using SPI protocol at high(er) frequencies

Serial Peripheral Interface (SPI) protocol is very successful today for chip to chip communications.

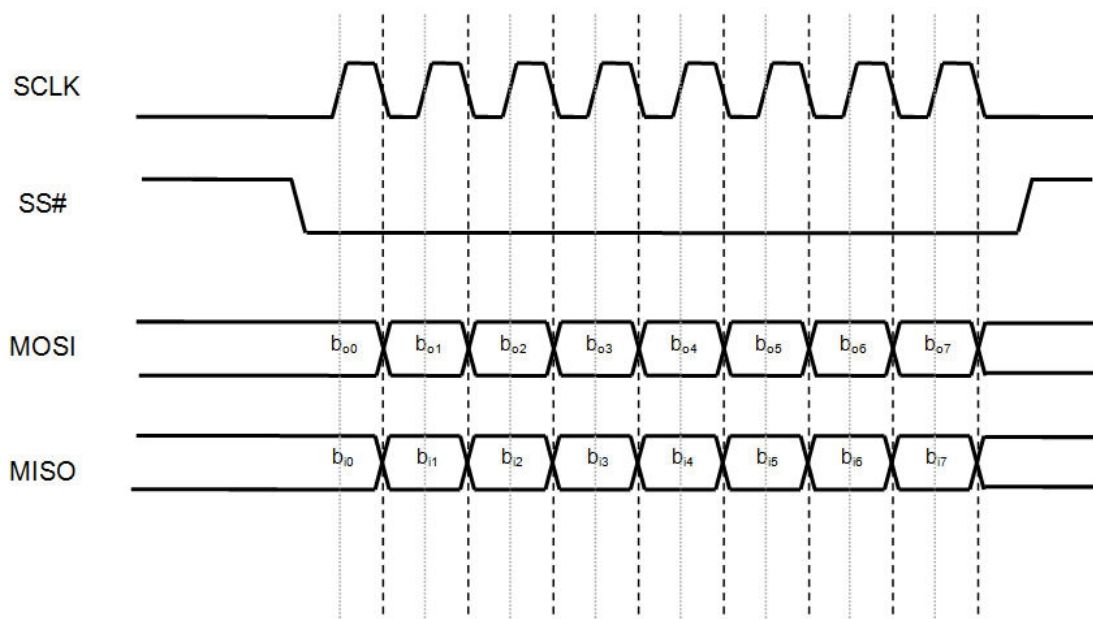
Our SPI Storm product is one of the most capable SPI host adapters on the market, with numerous options to 'tweak', upgrade and adapt the protocol to the user's needs. We sometimes get the feedback that this device has an excessively broad scope in the sense that most users really look for an adapter that supports 'just SPI', no more, no less.

There are multiple reasons for providing an extended range of options: SPI is not a so well defined protocol and many 'flavors' of SPI really coexist on the market today. Another example is that - unlike a protocol like I<sup>2</sup>C - SPI does not define the clock frequency to be used. Our experience shows that the usual frequencies for SPI are around 10 MHz or perhaps 20 MHz. With technologies like FPGA, it is rather easy to use clock frequencies that go up to 50 MHz or even 100 MHz.

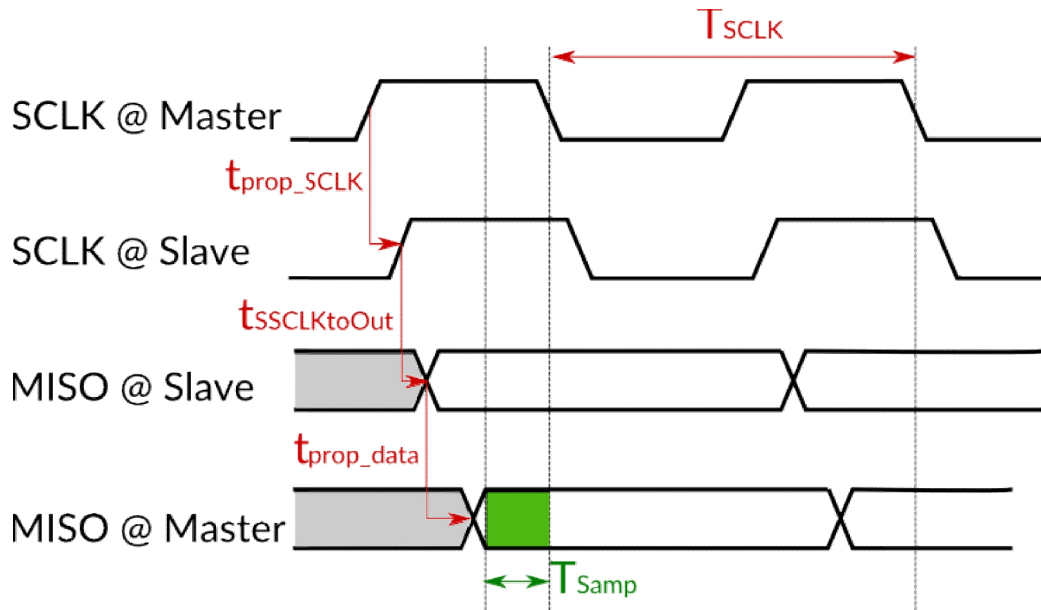
In this paper, we will show why testing a 100 MHz SPI requires controlling the protocol parameters.

## Serial Peripheral Interface (SPI) - an overview

SPI is fairly straightforward. It is a master-slave protocol scheme, with one single master on the bus. The Master initiates the communications (read or write) with the slaves and notably sends the reference clock signal to all the slaves. It uses at least 4 signal lines: SCLK (clock), MOSI (Master-Out-Slave-In), MISO (Master-In-Slave-Out) and one SS (slave select) line per slave. The master uses one clock edge to send data and the opposite edge in the same period to sample the data sent back by the slave.



The figure below depicts a read operation. This is the longest path from the master to the slave and back.



Symbol	Description
$T_{SCLK}$	SCLK period
$t_{prop\_SCLK}$	SCLK propagation delay from master to slave
$t_{SSCLKtoOut}$	Slave clock (SCLK) to output delay: delay between the clock edge at slave input and the data at the slave MISO output.
$t_{prop\_data}$	MISO Data propagation delay from slave to master
$T_{Samp}$	Timing budget available to the master for sampling the read back data.

The path starts at the Master's output. The clock signal SCLK is present at the output pin of the Master and is propagated on the (wire or trace) connection between the Master and the Slave.

The Slave responds by using this clock edge (e.g. the rising edge) to provide the requested data onto its MISO pin: this is 'MISO @ slave'. Between the reception of the SCLK signal at the slave's clock input and the moment when the requested data is present at the MISO output, there is a delay  $t_{SSCLKtoOut}$  ('Slave SCLK to data output').

The data on MISO is then propagated from the slave to the Master MISO input. The Master uses the falling edge of SCLK to sample this data<sup>1</sup>. We consider a standard SPI protocol, where data is generated on one edge and sampled on the opposite edge within the same clock cycle.

<sup>1</sup> In the reality, he can also use an internal clock signal, a delayed version of SCLK or a resynchronized version of SCLK. Of course, in such case, the timings relations between the SCLK signal at its input and the internal clock used for sampling the data must be taken into account.

The green area shows the timing budget that is available to the Master for sampling the data. This budget must be sufficient to allow the master to properly sample data, and depends on the characteristics of the master's I/O and technology.

Based on the timings labeled on the picture above, we have the following simple expression:

$$T_{\text{Samp}} = T_{\text{SCLK}} / 2 - t_{\text{prop\_SCLK}} - t_{\text{SSCLKtoOut}} - t_{\text{prop\_data}}$$

## Putting some numbers in ...

**Example 1:** SCLK interface at 4 MHz. Master and Slave are interconnected with simple 15 cm wires that have a propagation delay of 100 ps per centimeter. The wires on the SCLK and the MISO are identical. The slave has got a 6 ns clock to output delay.

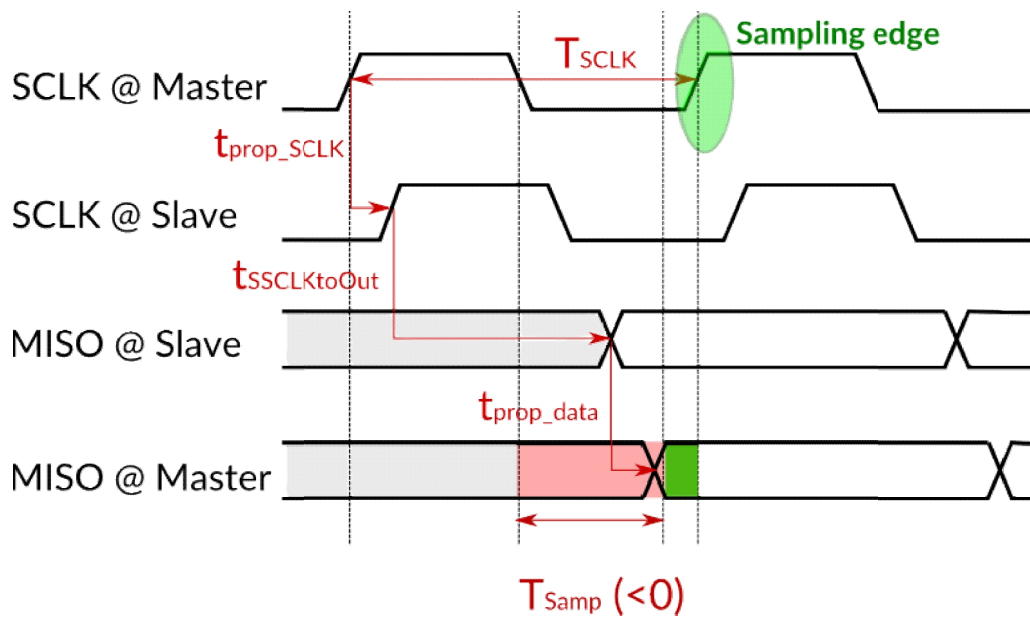
Symbol	Value
$T_{\text{SCLK}}$	250 ns (= 1/(4 MHz))
$t_{\text{prop\_SCLK}}$	1.5 ns (= 15 x 100 ps)
$t_{\text{SSCLKtoOut}}$	6 ns
$t_{\text{prop\_data}}$	1.5 ns
$T_{\text{Samp}}$	$250 / 2 - 1.5 - 6 - 1.5 = 116$ ns

No problem here: there is plenty of margin, even with a slow master requiring - e.g. 5 ns of setup time.

**Example 2:** SCLK interface at 100 MHz. Master and Slave are interconnected with simple 10 cm traces that have a propagation delay of 64 ps/cm.

Symbol	Value
$T_{\text{SCLK}}$	10 ns
$t_{\text{prop\_SCLK}}$	0.64 ns
$t_{\text{SSCLKtoOut}}$	6 ns
$t_{\text{prop\_data}}$	0.64 ns
$T_{\text{Samp}}$	$10 / 2 - 0.64 - 6 - 0.64 = -2.28$ ns (!)

The available budget is negative as seen from the waveforms:



So the transition on MISO occurs 2.28 ns *after* the SPI conventional sampling edge. The higher interface frequency does not allow to use a standard SPI protocol because of the total timing budget is insufficient for the selected connections and slave characteristics.

**Which options are there to ensure proper sampling at this speed?**

**Option 1:** Shorten the connections and find a Slave with a shorter clock to output delay. However such a 'faster slave' may simply not be available. Using shorter or 'faster' connections will also have a very limited impact because this is the smallest share of the 'missing' timing budget.

**Option 2:** Slow down the bus until sufficient margin is available. If the Master requires at least 2 ns of setup time to properly sample the data. We can deduce the bus maximum frequency of operation:

By reversing the expression and replacing  $t_{Samp}$  by the minimum setup time ( $t_{setup}$ ):

$$T_{SCLK} \geq 2 \times (t_{setup} + t_{prop\_SCLK} + t_{SSCLKtoOut} + t_{prop\_data})$$

$$T_{SCLK} \geq 2 \times (2 + 0.64 + 6 + 0.64) = 18.56 \text{ ns}$$

This means that the SCLK frequency can be at max:  $1/ 18.56 \text{ ns} = 53.87 \text{ MHz}$  if a 'pure SPI' protocol has to be used with the provided system.

**Option 3:** Use another master edge for sampling data. As seen on the diagram above, the next rising edge probably has sufficient timing margin to sample the data. Calculating the timing budget for this edge:

$$T_{Samp} \text{ (Next rising edge)} = T_{SCLK} / 2 + T_{Samp}$$

$$T_{Samp} \text{ (Next rising edge)} = 5 - 2.28 = 2.72 \text{ ns}$$

The next rising edge can indeed be used for sampling if, for instance, the Master needs at least 2 ns of margin to sample data.

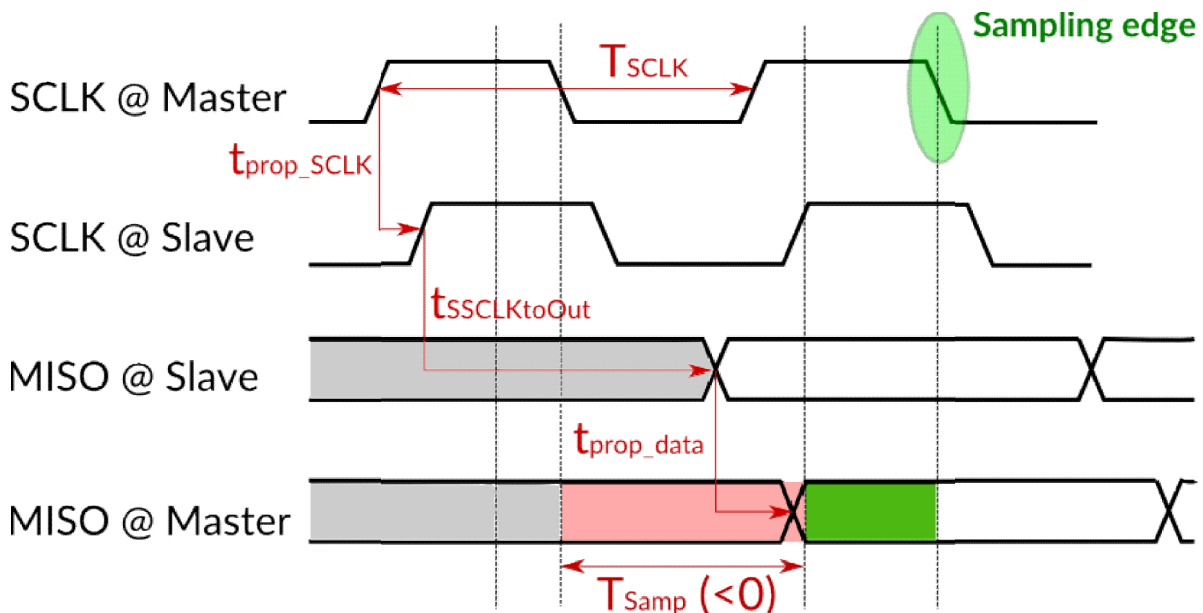
This option should be really considered as viable. The purpose of going for higher bus speed is all about getting a larger bandwidth for transmitting data. There are many components like memories, DACs and ADCs on the market today that already 'tweak' the protocol slightly in order to get more total throughput.

## Going further

Consider a very poorly designed system, with large propagation delays on the wires and a 'slave clock to output' as large as 8 ns. We keep a 100 MHz working frequency. The clock to output delay eating up 80% of a clock period will certainly create problems for a 'pure SPI' protocol.

**Example 3:** SCLK interface at 100 MHz. Master and Slave are interconnected with simple 15 cm traces that have a propagation delay of 100 ps per centimeter. Slave 'clock to output' is 8 ns.

Symbol	Value
$T_{SCLK}$	10 ns
$t_{prop\_SCLK}$	1.5 ns
$t_{SSCLKtoOut}$	8 ns
$t_{prop\_data}$	1.5 ns
$T_{Samp}$	$10 / 2 - 1.5 - 8 - 1.5 = -6 \text{ ns (!)}$



From the picture above, it can readily be seen that the data can't be sampled back by the clock falling edge, as it is usually defined for SPI, but, in addition, the data is ready for sampling at the input of the master only 6 ns after this edge, i.e. 1 ns after the next rising edge of the clock. The first clock edge available for sampling at the master input is the next clock period's falling edge, with a margin as large as 4 ns.

## Conclusions

Keeping a sufficient timing margin for sampling data on a SPI interface is not an easy task when the interface's clock is in the 100 MHz range.

The 'standard SPI' interface protocol which specifies generating data on one edge of the SCLK and sampling data on the opposite edge within the same period is actually a tough constraint at higher speeds.

But using a faster clock is desirable in order to reach higher data throughput using a serial protocol, and the technologies available on the market today are well able to handle such speeds.

For this reason, many serial protocol 'derived' from SPI are commonly used today. One of the main solutions for being able to use clock frequencies as high as 100 MHz consists in sampling readback data during the next clock period in order to give sufficient time to the Slave to respond - and still keep a communication scheme where the reference clock is provided by the Master.

## About SPI Storm

Byte Paradigm's SPI Storm is the most advanced USB serial protocol host adapter available today. It is used for functional testing on systems using SPI and many other serial protocols.

Unlike its competitors, **SPI Storm goes up to 100 MHz** and notably **includes a custom protocol definition engine** that provides you **full control** of when you want to sample data from any SPI(-like) slave. Hence, it can easily handle the situations described in this paper.

For more information about SPI Storm, please go to: <http://www.byteparadigm.com/products/spi-storm/>