Test MIPI DSI protocol conformance

Steve Tatarczuk - January 09, 2013

The MIPI (Mobile Industry Processor Interface) Alliance specifies series of interface specifications that can be used in mobile devices. One goal of MIPI is to provide a standard that allows components from different manufacturers to communicate, given they all adhere to MIPI standards. This eases the difficulty in using components from many different suppliers in one device. A second goal of MIPI is to reduce power consumption in order to increase battery life of mobile devices; the physical layer of MIPI utilizes low power consumption methods for communication.

Currently, the two most widely implemented uses of MIPI are the CSI-2 and DSI standards, which are the protocol specifications for communication between the host processor and cameras (CSI-2) and displays (DSI). This post will focus on DSI testing.

To test DSI at the UNH-IOL, the P331 signal generator by The Moving Pixel Company is usually used to play the part of a host processor and generate DSI protocol commands. The signal generator is connected to a probing board that allows an oscilloscope to sample the data being sent to and from the device. The signal continues through the probing board to the display being tested; the probing board is also used to pick up response signals from the display in the event that it is a bi-directional device.

Test Specifics

The commands that are sent to displays are defined in the UNH-IOL DSI Conformance Test Suite. The test suite defines tests of DSI functionality, such as the basic ability to interpret DSI protocol packets and certain error detection capabilities expected of bi-directional devices. Testing for the expected error detection capabilities (and implementing them in a device) is often the most difficult process. An example of a test is as follows:

“Configure the testing station acting as a host to transmit a write command with a WC field that does not match the number of payload bytes, this should not cause a CRC error to be detected by the DUT. Perform the test by transmitting a packet with a WC that is less than the payload length, a WC that is greater than the payload length, and a short packet with a long Data Type code. Verify that in each case the DUT transmits an Acknowledge and Error Report to the Host at the next opportunity with the Invalid Transmission Length bit set”
In this test, a packet is sent to the display in which the part of the packet that specifies how many bytes the device should expect to process (WC or word-count field) does not actually match the number of bytes transmitted in the payload. It is expected that the device being tested will not detect a CRC error (error stemming from the word count not matching the number of bytes received). Instead, a packet should be sent back to the host processor at the next available time period with an “Acknowledge and Error Report” packet with the Invalid Transmission Length bit set to “1” which indicates that the display detected an Invalid Transmission Length. For this test, the scenario where the word count does not match the length of the payload is as one of the following conditions:

- The word count tells the device to expect a payload that is smaller than the payload that is actually delivered.
- The word count tells the device to expect a payload that is longer than the payload that is actually delivered.
- The part of the packet that specifies what type of data the packet consists of (a long packet, short packet, acknowledge and error report packet, etc.) tells the device that it is a short packet but it is actually a long packet.

If a packet with the wrong word count gets through undetected, the device will expect the packet to end after it has read a certain number of bytes. If, for example, the word count was too small in a packet “X”, and the device did not recognize the error, it would stop interpreting the packet and begin on the next packet, “Y”, before it was actually done reading “X”. In this case, “Y” would actually still be “X” because it stopped reading “X” too early, but the device would not be aware of this, and it would start reading in the middle of the payload of “X,” expecting information that would be at the start of a new packet. In an application such as a display, this would ultimately mean that the image or video would not display properly.

The DSI specification requires that devices be able to recover from faults caused by contention. This means that, at some point, both the processor and the display attempt to send information over a lane, meaning neither signal can be understood. After this point, a device will be unable to understand any further commands even if they are sent without contention occurring. If the device is reset, however, it is expected to be able to understand commands again.

**Getting It Done**

It can be difficult to create the above scenario because most devices and the DSI specification itself include methods to avoid the occurrence of contention. For example, when a host processor wants to receive a packet such as an error report from another device (the processor in DSI is the “master” and the display is the “slave”), it will send a Bus Turn-Around (BTA) signal, which indicates to the slave device that the master is not going to send any more information over the lane, so it is safe for the slave to use the line to send information.

When a BTA is sent by normal means, the master end of the lane is aware that it just sent a BTA so it will, as expected, not send information for a set amount of time so that the information from the slave end can get through. The contention test can be performed by using scripts that tell the master side of the lane to send delays of certain voltages to manually create a signal that looks like a BTA in a way to “trick” the processor into telling the display it is safe to send packets even though a “real” BTA was not actually sent and the processor will still be sending packets when the slave side starts to send packets, causing them to collide and cause contention.

The means of creating the necessary scenarios vary from one device to another. Some tests can be performed on many devices easily and some, like causing certain types of contention and testing
how the device responds, can be more difficult. Once the desired scenario has been created and the information can be seen on the oscilloscope, the waveform is captured and is decoded in DPHYGUI, a useful program created by the UNH-IOL's Andy Baldman, which can decode DSI packets (as well as CSI-2 and perform other MIPI related analysis). Once decoded by DPHYGUI, it is simple to see whether the relevant requirements have been met based on the information that has been captured by the oscilloscope. Other tests require only a screen shot of relevant data on the oscilloscope or only indicate pass or fail with images on the display itself.

About the Author
Stephen Tatarczuk is MIPI Technician at the UNH-IOL.

Related Articles
Scopes decode MIPI UniPro and LLI protocols
MIPI battery interface charges up
Reduce Mobile Device Costs and Board Area with MIPI Low Latency Interface (LLI) and M-PHY