1/22/2018

Summary (MAC FW tests at MSU)

The main focus was to perform the timing adjustment, the Rx clock vs RX data. Two methods were tested. The first method assumes to correct the timing using the I/O delay block (IDelay3 and ODelay3 component) on the FPGA. But a closer look into the Vivado files leads to the following conclusions that only a RX data path in the MAC design uses a Input delay block. The RX clock path uses the IBUF and BUFG only and the output from BUFG is wired to the IDDR. Unfortunately the MAC IP is protected (read-only mode). Any kind of modification might be impossible. There is a trick to modify the contents and put it back to the Vivado but unfortunately the internal protection ignores this.

That means, we can practically tune the RX data delay only. The Input delay block on the UltraScale allows us to delay the data by 1.25 ns (0 - 1250 tap, each tap corresponds to a 1ps).
In principle, changing the delay value in generic block (in vhdl) or in xdc file should allows to achieve the goal. But two problems discovered here: There is a static delay for the RX data in the mac.xdc file. That is probably fine, since that delay corresponds to a RX clock route delay on the FPGA.
But this file has read-only mode. It can be changed using an external editor and dropped again into the Vivado environment but Vivado might ignore it.

There is a way to define the order which is being used to process the xdc file in Vivado. So, this way we can overwrite the delay value written in mac.xdc file.
However, some specific delay values for the RX data introduces some timing closure issue.
Of course, this can be fixed by running the Vivado timing analyzer which provides the details for which path the problem with the timing occurs (yes. It's fixable).

All this leads to the conclusion that the timing adjustment in FPGA (using I/O block) in that particular case is somehow possible (Input delay block is implemented for the RX data only) but uncomfortable.

The second method which we explored relies on the timing adjustment on the PHY chip. In this case, the MAC FW uses the default Xilinx timing settings and that corresponds to the RX data delay (0.9ns) only (to compensate the RX clock route delay).

We conducted a timing scan, the starting point assumed the RX clock is delayed by +0.96ns (Phy chip) while the RX data speeded up by -0.42 ns (Phy Chip). That MAC FW design was not tested in that timing regime before. We conducted a several runs by adding the delay to the RX data.

In the region for the RX data delay (-0.42 to 0ns), we noticed that when we transmit 5 packages from Ostinato to a MAC FPGA, we receive back 5 packeges (from MAC FPGA) in Wireshark. That result is good but the received data length doesn't correspond to a source data length. In addition to this, when we look at the received data, the data contents looks different than the source data.
[We need to note that if everything works as expected we should be able to move the RX clock by ca. 0.5 ns and still see a correct data].

The current assumption is that the contents we receive from the MAC FPGA should look the same as the source data. We conducted a few more tests. We monitored the RX data (4 bits) and the data valid flag on the FPGA directly from a Phy chip.
The result is that we can capture these bits with the chipscope and somehow recognize the contents. In a few more runs we change the source data contents (a few bits), and that change is visible on the chipscope too.
There is one aspect that which needs to be taken into account. The chipscope has its internal scheme to probe the data. And, we need to fulfill that criteria to grab a good data contents.

In the final step, we lower the speed from 1Gbps to 100Mbs. We also modified the FW to be connected to the Phy chip (U21).
In this configuration, we conducted a several tests. Dan monitored the 25MHz clock from the Phy chip on the scope. In addition to this, Dan monitored the RX data (+DV flag) contents on the scope (Mac address source and destination). If we see these data on the scope it must work!

I tried to monitor these data on the chipscope (as I mentioned above the chipscope probes the data using the specific scheme). I captured several packages but the Vivado hardware manager complains that the data contents is corrupted. It provides some hints and that needs to fixed to capture a good data contents. I need to fix it.

The conclusions:

1. (1Gbs) With the best timing settings we can send the data (5 packages) from Ostinato to the MAC FPGA, and receive this data back (5 packages).
The data contents looks rather incorrect (both the data contents and data length).

2. (1Gbps) We can monitor the RX data directly from the Phy chip. I believe that by the RX clock adjustment we should see a valid package on the chipscope.
Some philosophy is needed here to probe these data. We were close but some tuning is needed to see a correct package.

3. (100Mbs) Currently, I am working to capture the 4 b RX data (directly from the Phy chip) on the chipscope. That part needs be done as first test. Once we see the a proper data contents (RX data) on the chipscope, this is practically 50% of success.

4. (100Mbs) I will provide the MAC FW for the HUB module from scratch. And, we start from 100Mbps because we practically eliminate the timing scan needs (RX clock vs RX data).