USB 3.1 PHY Implementation

Amr Elhosiny

October 24, 2014

1 HDL Language

VHDL will be used for RTL coding. Verilog and/or VHDL will be used for test bench coding.

2 FPGA Device Family

Will use Xilinx Spartan-6 FPGA for speed and cost considerations. Also it has a low power category.

3 Package

A VHDL package will be created listing:

- All hard-coded packet types in the document will be defined with names to be easily compared and detected in the RTL.
- It will define needed functions.
- Packages and functions specific for test benches will be in separate packages.

4 Clock Generator

There will be a central clock generator module which generates divided clocks, gated clocks. It may instantiate FPGA specific modules like BUFGE's.

5 Packet Generation & Detection

There are many design alternatives listed below.

5.1 Packet Type Detector

Depending on packet type, some blocks or functions are enabled or disable. There will be a central packet type detector module with an active high output bit for each packet type. For packets that have common effect on loaded modules or functions, more possible outputs may be grouped into less or single output bit. The type of packet will decide the behaviour of the data path modules.

5.2 State Machine

Packet driver may be a state machine implementing some features of the link layer. It will transmit/receive certain sequences and implement limited link layer features. It will then produce required control signals defining the type of transmitted/received packet to control the behaviour of the datapath modules. Examples for the link layer features that may be implemented :

- A state machine will handle link initialization.
- Then sending data packet.

A set of assumptions will be made to simplify the state machines and the implementation for example implementing specific not all modes of operation.

5.3 Downstream & Upstream Ports

The host machine will implement extra features not present in the USB device.

6 Scrambler

Will have a toplevel consists of a scrambler core alongside with a controller. The controller will tell the core to enable or disable scrambling, advance the LFSR, XOR or do not XOR data.

6.1 Gen-1

6.2 Gen-2

7 Source Control

The implementation database is managed using git as source control. The git repository could be found in the URL `https://github.com/aelhosiny/usb3_phy`.

8 ToDo

- Define clock rates according to datapath width changes. All clocks will be generated from DCM's to guarantee the phase relation and domains.
- The core clock should meet two conditions:
  1. It should be related to the serial to parallel input rate with an integer relation.
  2. It should be related to all needed datapath clocks with and integer relation.

9 Prototype

The prototype will be implemented on two FPGA's. One board will act as TX, the other FPGA will act as RX. The two FPGA's will be completely asynchronous. Data payload and program will be sent to TX FPGA via one of the possible interfaces:

- USB-SPI interface.
- Serial interface.

- UART.

Some link layer functions will be implemented in hardware. They will be activated via a set of opcodes sent from the PC-Board interface.

Other possible implementation is to use xilinx micro blaze core to implement some link layer functions. This should implement a more complex solution.

The data out from TX will be serialized using a normal digital serializer just for prototyping. However, both Gen1 and Gen2 PHY's will be designed to meet the operational output rate to the SerDes. The clock frequency could be lowered later to meet prototyping requirements.