Hardware verification through software scheduling for USB using xHCI

Wasiq Zia
Navneet Jha
Vipin Chauhan

Problem Statement/Introduction

USB 3.0/3.1/3.2 usage has been increasing exponentially with time, be it in Mobile phones, computers or any other device.

Robustness, completeness and scalability of USB verification environment is needed to be strengthened for the current advanced usage at subsystem levels with low power, high throughput being the need of hour

Scalable verification component that is flexible enough to verify a USB Host Controller at IP, sub-system as well as system level is required

Proposed Methodology/Advantages

Test sequences driven from a higher abstraction layer, so that the same sequences can be used at IP/System Level.

Generic interface connecting the Host Controller Driver(xHCD), the system memory for data access and the Host Controller(xHC) was defined.

A translation layer to convert the memory-based transactions to protocol data units for easy interpretation of verification engineers.

Platform agnostic : Definition at driver level ensures that it can work with acceleration as well as simulation

Implementation Details/Diagram

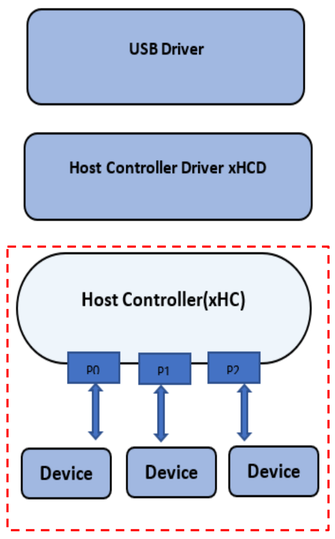
xHCD is the software part that creates relevant data or commands understood by xHC to perform relevant operations.

A generic register-based interface has been defined for programming.

Software managed memory which makes it easier to configure the setup and program sequences.

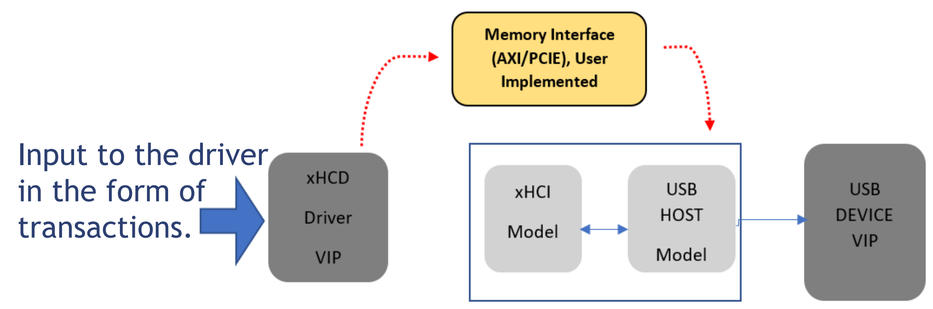
Option for internal and external memory instantiation

Callback based mechanism to connect any possible interface through a glue logic. Any memory protocol can be adapted to work between xHCD and Host Control



Implementation Details/Flow Chart

Generic Interface. For RTL and subsystems, various protocols like PCIe, AXI or AMBA are used for connecting these blocks, since the interface is not pre-defined in any of the USB specifications.



Driver creates packed data for TRB memory writes from the transactions and based on the memory map provided by user. Output available through callback to be written to DUT memory through AXI interface

Results Table

There have been issues in designing verification components for USB Host Controller which are tied to specific interface.

The proposed implementation creates an interface independent verification component while allowing for reuse across all levels of the system hierarchy.

It is flexible and robust enough to allow porting any existing host controller sequence library to this model

The existing USB3 sequences can be ported and scaled to work with tunneling requirements of sub-system protocols like USB4

The solution was used for systems employing PCIe, AXI as connection interfaces and seamlessly connected with both.

Conclusion

Tests written at xHCD level help mimic real system stimulus as full chip is being controlled through the driver.

It models real traversal of data in a system thus giving a clear picture of host controllers interaction with memories for packetization and control.

Great model for software controlled verification using classing design verification components. It can work with stimulus generated through high level languages like C++ as well as verification languages like Verilog etc.

It gave us flexibility not just within the USB stack but also allowed for re-using the same environment for cross protocol interactions with subsystems that consume USB.

For e.g. tunneling the USB3 traffic through USB4 model. It is easier to control traffic through the higher layers. It allows interaction at controller level.

REFERENCES

The USB 3.2 Specification released on September 22, 2017 and ECNs | USB-IF

eXtensible Host Controller Interface for Universal Serial Bus (xHCI) (intel.com)