Verification of USB 2.0 Functional Core

Parth R. Zalawadia [1]
[1] M. E.
[1]VLSI & Embedded System Design
[1]GTU PG School Gujarat Technological University, Ahmedabad, Gujarat, India

Abstract---A verification environment to verify USB 2.0 Functional core using System Verilog is implemented in this paper. The verification IP can be reused to verify any USB protocol. Verification Environment is built to check the behaviour of DUT and to check whether we meet desired functionality of the core so that we can cover all the aspects of the design during verification. Verification of UTMI, Protocol Layer, Wishbone Interface and Memory Arbiter block is implemented in this paper. The Simulation Results are generated using QuestaSim 10.0b.

Keywords: Verification Environment, System Verilog, USB 2.0

I. INTRODUCTION

The Universal Serial Bus (USB) has evolved to the standard interconnect between computers and peripherals. Everything from a mouse to a camera can be connected via USB. With the new USB 2.0 specification, data rates of over 480 Mb/s are possible. The Universal Serial Bus is a point to point interface. Multiple peripherals are attached through a HUB to the host. This core provides a function (peripheral device) interface. This core fully complies with the USB 2.0 specification and can operate at USB Full and High Speed rates (12 and 480 Mb/s). [1]

SystemVerilog is a special hardware verification language to be used in verification. SystemVerilog provides high-level data structures and the notion of dynamic data types. SystemVerilog provides an object-oriented programming model. In this Paper, components of verification environment are implemented for UTMI, Protocol Layer, Wishbone Interface and Memory Arbiter block.

II. ARCHITECTURE OF USB 2.0 FUNCTIONAL CORE

In this Paper, for verification of USB core drawn an USB 2.0 design from opencores.org and the architecture is shown in Fig.1. The USB 2.0 Functional Core consists of 6 modules: Protocol Layer Module, UTMI Module, Memory Arbiter Module, SSRAM module, Physical layer interface module and Wishbone module.

UTMI block consist of Interface State Engine and Speed Negotiation Engine. The Interface state engine block tracks the interface state. It controls suspend/resume modes and Full Speed/High Speed switching. An internal state machine keeps track of the state and the switching of the operating modes and the Speed Negotiation block negotiate the speed of the USB interface and handles suspend and reset detection. [1]

Protocol Layer block consist of packet assembly, packet disassembly, and protocol engine block. Packet assembly block assembles packets and places them in to the output FIFO. It first assembles the header, inserting a proper PID and check sums, then adds a data field if requested. Packet Disassembly block decodes all incoming packets and forwards the decoded data to the appropriate blocks. The decoding includes extracting of PID and sequence numbers, as well as header check sum checking. Protocol engine block handles all the standard USB protocol handshakes and control correspondence. Those are SOF tokens, acknowledgment of data transfers (ACK, NACK, NYET), replying to PING tokens. [1]

The memory interface and arbiter arbitrates between the USB core and host Interface for memory access. This block allows the usage of standard single port Synchronous SSRAM. [1]

The SSRAM is a single ported Synchronous SRAM block that is used to buffer the input and output data. [1]

The host interface block provides a consistent core interface between the internal functions of the core and the function-specific host or micro controller. The host interface is WISHBONE SoC bus specification Ver. compliant. [1]

The Physical layer to complete the data transmission.

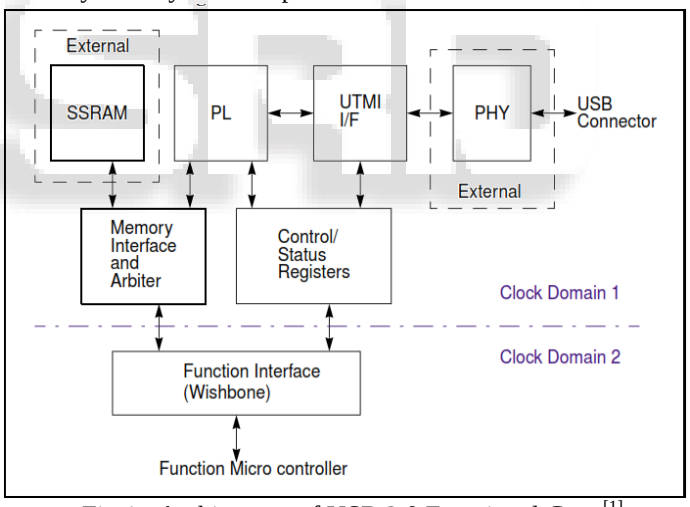

Fig.1: Architecture of USB 2.0 Functional Core [1]

III. VERIFICATION ENVIRONMENT

The purpose of Verification Environment is to generate stimulus with the help of stimulus generator, which are sent to DUT (Design Under Test) by driver. And scoreboard compares the actual and expected output to verify that the function is correct.

IV. INTRODUCTION OF VERIFICATION ENVIRONMENT

Fig.3 shows the verification environment. The Environment include following components: Generator, Driver, Monitor, Scoreboard and DUT written by System Verilog language, System Verilog test bench which include interface, top module, and test program.

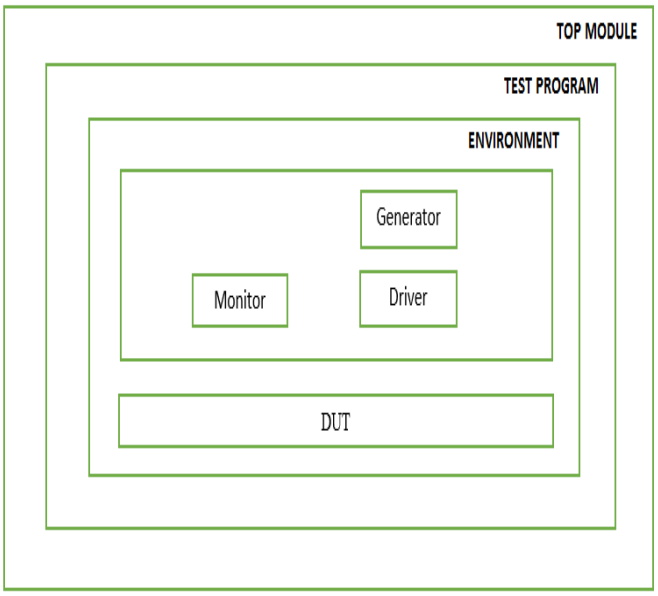

Fig. 2: Overview of Environment

In System Verilog test bench the generator component generates stimulus which are sent to DUT by driver, and then the driver translate the operations produced by the generator into the actual inputs for the design under verification. The Monitor reports the protocol violation and converts the pin level activities in to high level. Also the results can be checked by the scoreboard.

The fig.1 is divided into three parts of the verification environment. The first is the DUT, which is the design top module to be verified. The second is the system verilog test bench. The third is the interface part used to joining the DUT and the System Verilog test bench. All this parts are instanced in the test top.

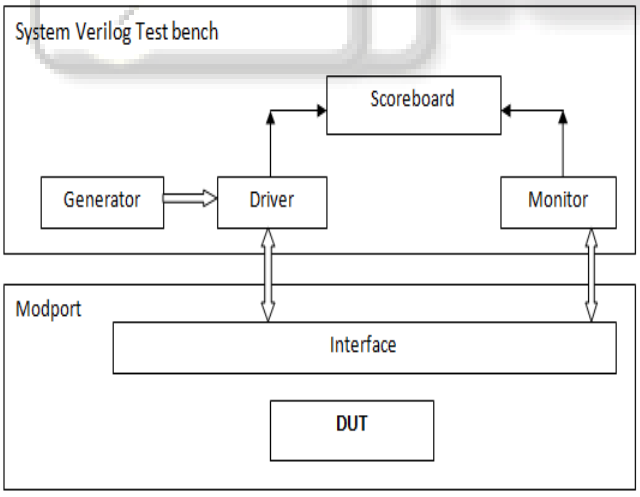

Fig.3: Verification Environment

V. SIMULATION RESULTS

A. Simulation of Wishbone Interface

It first transmits data by using the WISHBONE master port's function. When started working, the system is reset by the reset signal rst, when wb_stb_i and wb_cyc_i signal asserts it determines an effective transmission cycle by sending wb_cyc_i and wb_stb_i signals to WISHBONE interface.

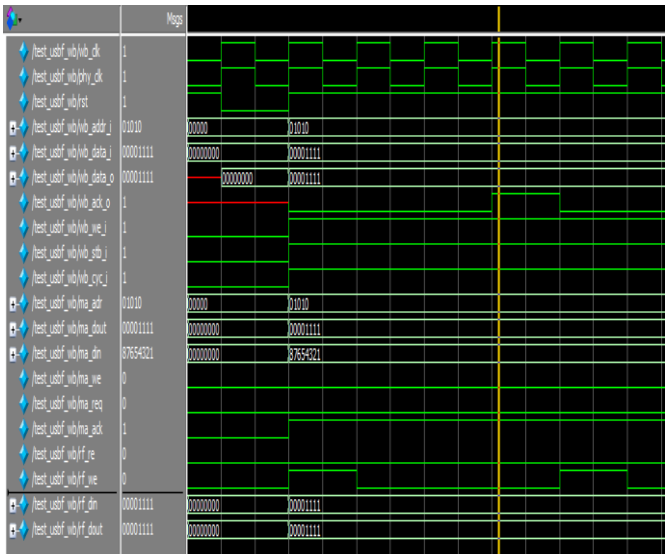

Fig.4: Wishbone Interface

According to wb_we_i level, the system determines whether the data is reading or writing. When wb_we_i is high, main port writes data to WISHBONE interface, but when wb_we_i is low, port reads out data from WISHBONE interface. Data Communication rely on answering signal wb_ack_o: at the rising edge of the wb_ack_o signal the main port reads out data from data bus or writes data to data bus.

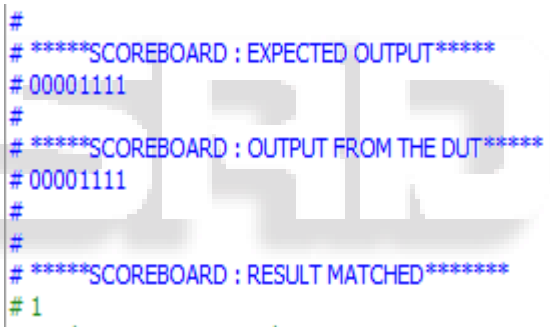

Fig. 5: Scoreboard Output of Wishbone Interface

B. Simulation of UTMI Interface

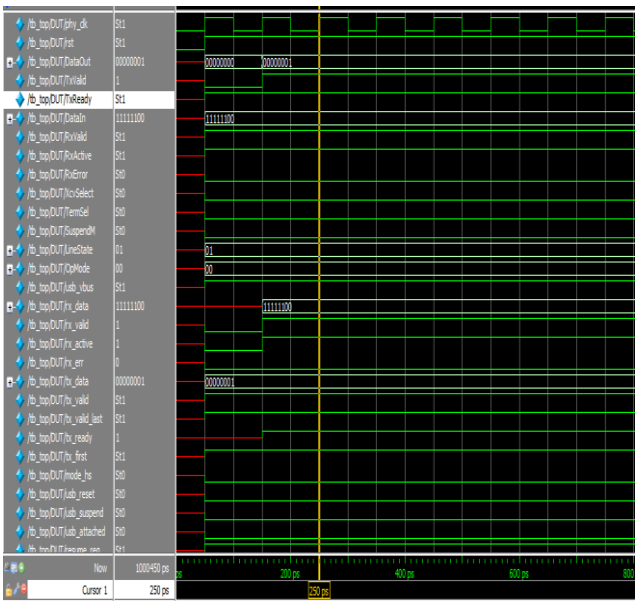

Fig.6: UTMI Interface

Initially when started working, the system is reset by the reset signal rst, when resume_req signal and rx_Active signal is asserted then it transmit the data. It will check data was valid or not by Rx_valid signal. It will also check tx_ready signal that transmitter is ready to receive data .Here Xcvr_select signal is low which determines FS transceiver is enabled. SuspendM signal is low which determines Macro cell circuitry drawing suspended current. LineState signal is 01 which determines J state is selected so value of DP and DM is 0 and 1 respectively. Here the OpMode signal is 00 which determines core is operating in normal operating modes. Mode_hs signal is low which determines USB core is currently working in FS mode.

```
#
# *****SCOREBOARD : EXPECTED OUTPUT*****
# fc
#
# *****SCOREBOARD : OUTPUT FROM THE DUT*****
# fc
#
#
# *****SCOREBOARD : RESULT MATCHED*******
# 1
```

Fig. 7: Scoreboard output of UTMI Interface

C. Simulation of Packet Assembly

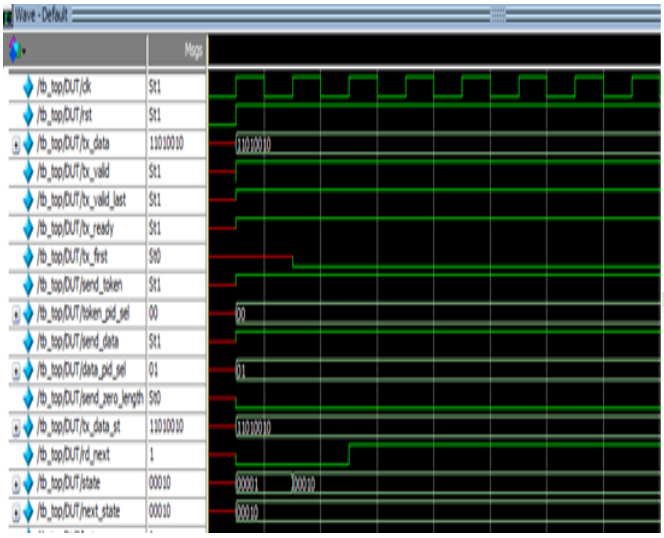

Fig.8: Packet Assembly

Initially System is reset by the rst signal. The first step is to send the token and then data when tx_ready signal is asserted. Here corresponding token_pid_sel signal and data_pid_sel shows 00 and 01 which determines if it is 00 then token packets and if 01 then data packet. The second step is to check whether the token and data is valid or not using tx_valid signal. It also checks whether last token and data are valid or not using tx_valid last. And finally after completion of cycle it will read next token and data by asserting rd_next signal.

```
# *****SCOREBOARD : EXPECTED OUTPUT*****
# d2
#
# *****SCOREBOARD : OUTPUT FROM THE DUT*****
# d2
#
#
# *****SCOREBOARD : RESULT MATCHED*******
# 1
```

Fig. 9: Scoreboard output of Packet Assembly

D. Simulation of Packet Disassembly

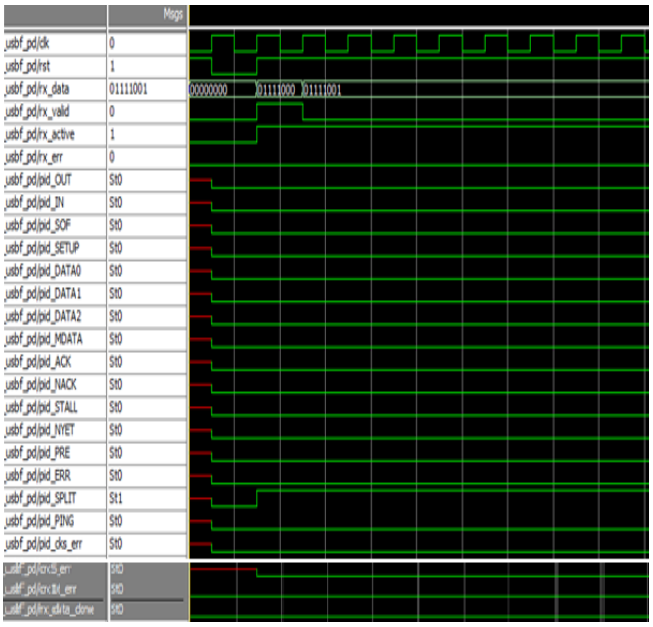

Fig.10: Packet Disassembly

When started working, the system is reset by the reset signal rst. Initially it receives the data when rx_active signal is asserted. Here rx_valid is high which determines that data is valid. Signal rx_err is low which determines there is no error in the packet. Now it will decode all the packet and checks if it is token packet, data packet, handshake packet, special packet. Here pid_SPLIT signal is high which indicates it is special packet.

```
# *****SCOREBOARD : EXPECTED OUTPUT*****
# f8
#
# *****SCOREBOARD : OUTPUT FROM THE DUT*****
# f8
#
#
# *****SCOREBOARD : RESULT MISMATCHED*******
# 1
```

Fig. 11: Scoreboard output of Packet Disassembly

E. Simulation of Memory Arbiter

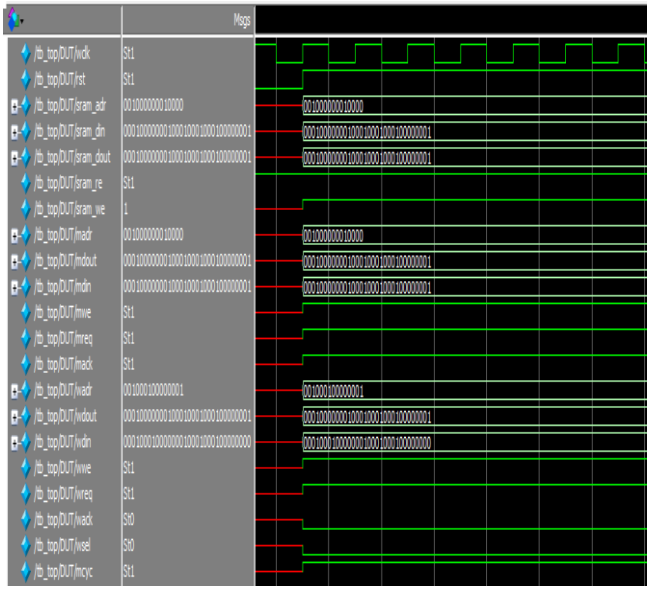

Fig.12: Memory Arbiter

In memory arbiter simulation result, there are three sub blocks: SSRAM, Memory and Wishbone. Fig.7 shows the output of both memory block and wishbone block act as input to SSRAM so both will send req signal at the same time to sram to write their address and data at SSRAM location, but out of both block sram will select one block to process at a time. Here sram selects memory block to transfer their address and data. Here mwe signal is high which determines memory writes data and address to sram, and also sram_re signal is high which determines it reads the data and address from memory.

```
#
# *****SCOREBOARD : EXPECTED OUTPUT*****
# 10111101
#
# *****SCOREBOARD : OUTPUT FROM THE DUT*****
# 10111101
#
#
# *****SCOREBOARD : RESULT MATCHED*******
# 1
```

Fig. 13: Scoreboard output of Memory Arbiter

VI. CONCLUSION

The individual components of the verification environment have been written and verified using System Verilog Language and by performing verification of USB 2.0 using System Verilog we have cover all the test cases and seen the behaviour of our system and we have met the functional requirements of the core. And cover each and every aspects of the design during verification. The simulation has been successfully carried out.

ACKNOWLEDGEMENT

The submission of the paper gives me an opportunity to convey my gratitude to all those who have helped me in completion of my paper. I hereby would like to acknowledge the invaluable support of my project guide Mr. Ashish Prabhu, Staff Engineer at LSI Corporation Ltd Pune, and have been an unending source of inspiration for me.I am also thankful to other Technical Assistant members who have directly or indirectly helped me whenever it was required by me.

I owe my special thanks to, friends for supporting and encouraging me in all possible ways and also all faculty members of the Centre for Development Advanced Computing (C-DAC)and Gujarat Technological University (GTU) for providing facilities to access to IEEE Library which helped me in findings of my project.

REFERENCES

[1] Universal Serial Bus Function IP Core rev. 1.5 January 27, 2002

[2] Ku, Yi-Nan; Yang, Yu-Sen; Han, Yang, “Device inter-face protocolimplementation based on USB2.0", Jilin Daxue Xuebao (Gongxueban), vol. 3 pp. 170-173, March 2005

[3] USB 2.0 Transceiver Macro cell Interface (UTMI) Specification, version 1.05, March 29, 2001

[4] LIN Huaiqi, CHEN Yong, “Design and Implementation of USB2.0 Device Controller IP Soft Core”, *Ship ElectronicEngineering*, Vol. 28, No. 3. Wuhan China, March 2008, pp.151-153,183.

[5] N.Bombieri, F. Fummi, "Hybrid, Incremental Assertion-Based Verification for TLM DesignFlows", *IEEE Design & Test of Computers*, 2007, vol. 24, no. 2, pp. 140 - 152.

[6] BAI Xiaoping, WEI Yuanfeng, “Fault-Tolerant Design and Testing of USB2.0 Peripheral Devices IP Core System”, *Tsinghua Science and Technology*, Vol. 12, No. S1. BeijingChina, July 2007, pp. 197-201.

[7] RUAN Lihua, Wang Xiang, HUANG Quanping, etc, “the Design and Development of USB2.0 Interface IP Core”, *Journal of Fudan University (Natural Science)*, Vol. 44, No. 1. ShanghaiChina, February 2005, pp. 173-177.

[8] Habibi and S. Tahar, “Design and Verification of SystemC Transaction-Level Models”, *IEEE Trans. on VLSI Systems*, 2005, vol. 14, no. 1, pp. 57 - 68.

[9] YIN Baqun, ZHOU Xianzhong, “Design and Verification of USB Protocol Layer Based on FPGA”, *China Integrated Circuit*, Beijing China, July 2008, pp. 37-41.

