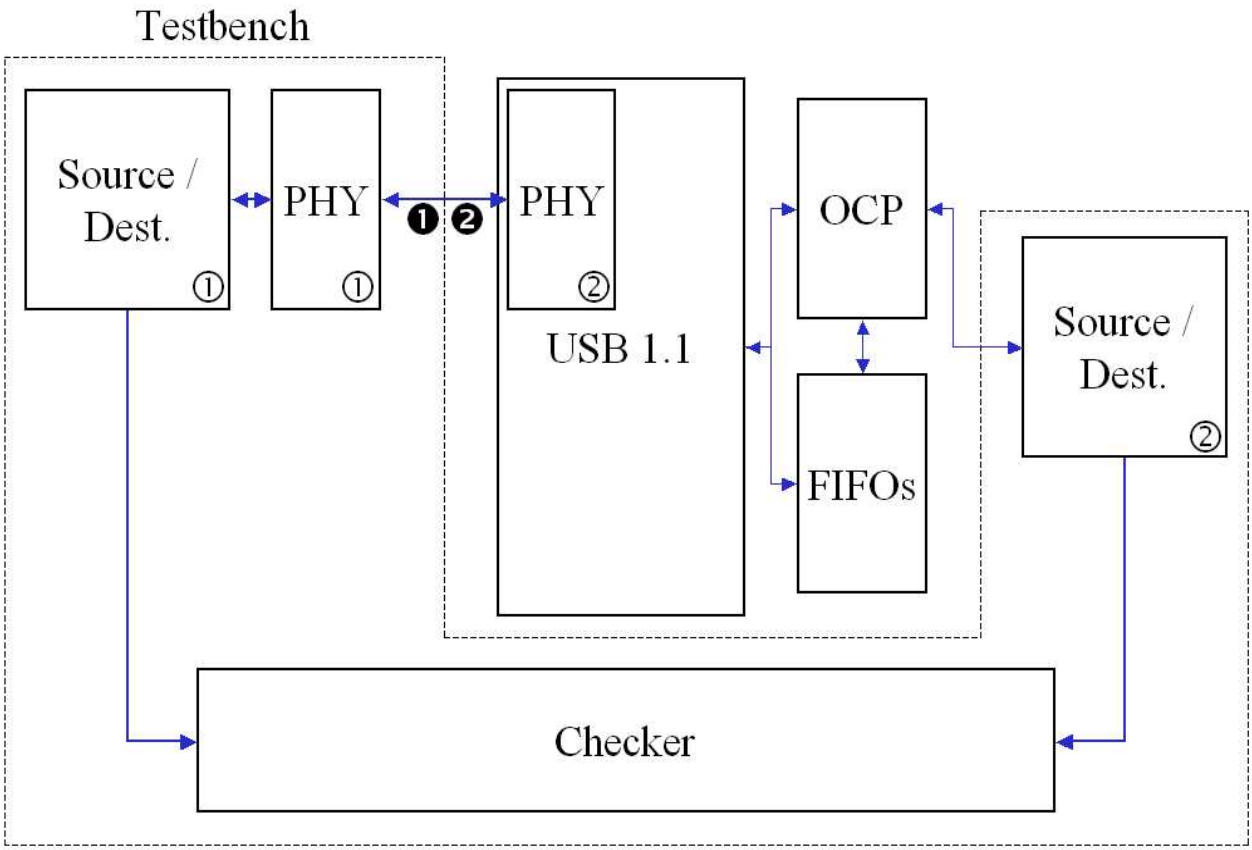
USB1.1 Testbench Documentation

(Module: “usb_ocp_test”)

The diagram below shows as the testbench is implemented:



1 – HOST
2 – FUNCTION

1 – HOST TRANSCEIVER
2 – FUNCTION TRANSCEIVER

Source/Dest. generates and receives packets, while that the Checker verifies if the packets sent of a side arrive equal of the other side. The transceivers are substituted by a direct connection (see “rx1_update” and “rx2_update” methods). Two functions (“utmi_recv_pack” and “utmi_send_pack”) write directly in the HOST PHY signals.

The Checker does the following verifications:

- The packet PID is the expected?
- The PID *check field* is correct?
- The packets order is in complaince with the protocol?
- The packet length is the expected?
- The packet data is the expected?
- The packet CRC is correct?

It follows a brief explanation of each testbench function:

- show_errors – it shows how many errors had occurred until that instant.
- crc5 – it calculates the CRC5 for packets of the following types: TOKEN or SOF.
- crc16 – it calculates the CRC16 for packets of the type DATA.
- utmi_send_pack – it plays the role of transceivers. It writes in the HOST PHY signals.
- utmi_recv_pack – it plays the role of transceivers. It reads of the HOST PHY signals.
- recv_packet – it analyzes the content of the received through the "utmi_recv_pack" function packet. It analyzes the packet length, verifies the CRC16 and the PID check field.
- send_token – it sends, through the "utmi_send_pack" function, a TOKEN packet of the specified PID. It calculates the CRC5 through the "crc5" function and calculates the PID check field.
- send_sof – it sends, through the "utmi_send_pack" function, a TOKEN packet of the SOF PID with the specified frame number. It calculates the CRC5 through the "crc5" function and calculates the SOF PID check field.
- send_data – it sends, through the "utmi_send_pack" function, a DATA packet of the specified PID. It calculates the CRC16 through the "crc16" function and calculates the PID check field.
- send_setup – run a CONTROL TRANSFER: SETUP-> DATA0-> ACK<-. The packet fields are specified. It uses the following functions: "send_token", "send_data" and "recv_packet". It verifies the packets order, PID and length.
- data_in – run a IN TRANSACTION: IN-> DATA0/DATA1<- ACK->. The choice between DATA0 or DATA1 is fulfilled by the "setup_pid" signal. The packet length is specified. It uses the following functions: "send_token" and "recv_packet". It verifies the packets order, PID, length and data.
- data_out – run an OUT TRANSACTION: OUT-> DATA0/DATA1-> ACK<-. The choice between DATA0 or DATA1 is fulfilled by the "setup_pid" signal. The packet length is specified. It uses the following functions: "send_token" and "utmi_recv_pack". It verifies the packets order, PID and length.
- setup0 – run the USB enumeration. It sets the USB address and receives the descriptors. It uses the following functions: "send_setup", data_in" and "data_out". The ENDPOINT 0 is tested.
- in1 – run ISOCHRONOUS IN TRANSACTIONS: IN-> DATA0<-. This function does the following sequence: SOF-> IN-> DATA0<-. It receives packets with random DATA of the ENDPOINT 1 (ISOCHRONOUS IN). It receives 4 packets of each length. The length are the following values: 0, 32, 64, 96, 128, 160, 192, 224 and 256. The maximum packet length of this ENDPOINT is 256. It uses the following functions: "send_sof", "send_token" and "recv_packet". It verifies the packets order, PID, length and data.
- out2 – run ISOCHRONOUS OUT TRANSACTIONS: OUT-> DATA0->. This function does the following sequence: SOF-> OUT-> DATA0->. It sends packets to ENDPOINT 2 (ISOCHRONOUS OUT). It sends 4 packets of each length. The length are the following values: 0, 32, 64, 96, 128, 160, 192, 224 and 256. The

maximum packet length of this ENDPOINT is 256. It uses the following functions: “send_sof”, “send_token” and “send_data”. It verifies the packets order, PID, length and data.

- in3 – run BULK IN TRANSACTIONS: IN-> DATA0/DATA1<- ACK->. This function does the following sequence: SOF-> IN-> DATA0/DATA1<- ACK->. It receives packets with random DATA of the ENDPOINT 3 (BULK IN). It receives 4 packets of each length. The length are the following values: 0, 8, 16, 24, 32, 40, 48, 56 and 64. The maximum packet length of this ENDPOINT is 64. It uses the following functions: “send_sof”, “send_token” and “recv_packet”. It verifies the packets order, PID, length and data.
- out4 – run BULK OUT TRANSACTIONS: OUT-> DATA0/DATA1-> ACK<-. This function does the following sequence: SOF-> OUT-> DATA0/DATA1-> ACK<-. It sends packets to ENDPOINT 4 (BULK OUT). It sends 4 packets of each length. The length are the following values: 0, 8, 16, 24, 32, 40, 48, 56 and 64. The maximum packet length of this ENDPOINT is 64. It uses the following functions: “send_sof”, “send_token”, “send_data” and “utmi_recv_pack”. It verifies the packets order, PID, length and data.
- in5 – run INTERRUPT IN TRANSACTIONS: IN-> DATA0/DATA1<- ACK->. This function does the following sequence: SOF-> IN-> DATA0/DATA1<- ACK-> or SOF-> IN-> NACK<-. It receives packets with random DATA of the ENDPOINT 5 (INTERRUPT IN). It receives 4 packets of each length. The length are the following values: 0, 8, 16, 24, 32, 40, 48, 56 and 64. The maximum packet length of this ENDPOINT is 64. It uses the following functions: “send_sof”, “send_token” and “recv_packet”. It verifies the packets order, PID, length and data.
- out6 – run INTERRUPT OUT TRANSACTIONS: OUT-> DATA0/DATA1-> ACK<-. This function does the following sequence: SOF-> OUT-> DATA0/DATA1-> ACK<-. It sends packets to ENDPOINT 6 (INTERRUPT OUT). It sends 4 packets of each length. The length are the following values: 0, 8, 16, 24, 32, 40, 48, 56 and 64. The maximum packet length of this ENDPOINT is 64. It uses the following functions: “send_sof”, “send_token”, “send_data” e “utmi_recv_pack”. It verifies the packets order, PID, length and data.

It follows a brief explanation of each testbench method:

- rx1_update – it does HOST PHY and FUNCTION PHY direct connection.
- rx2_update – it does HOST PHY and FUNCTION PHY direct connection.
- watchdog – it updates the watchdog counter.
- wd_cnt_mon – it halts the simulation if the watchdog counter to expire.
- init – it initiates the simulation.