Test planning for Cooperative Design Lab

Project: AHB-lite/USB
Team: Te-Yu Hsin,
Wen-Bo Hung,
Hsuanling Lee
Date: 16 April 2023

Module tested	Feature tested	Test name	Criteria Num.	Description of test inputs	Description of expected outputs	Check-offs
Top Level	Receive PID OUT and data less than 64 bytes	tb_usb_receive_out_normal	2.2	Reading PID OUT and data less than 64 bytes	Send PID ACK to the host Rx_error = 0	
Top Level	Receive PID OUT and data with error	tb_usb_receive_out_error	2.2	Reading PID OUT and data with error (early eop)	Send PID NAK to the host Rx_error = 1	
Top Level	Receive PID OUT and data with over 64 bytes	tb_usb_receive_out_over64	2.2	Reading PID OUT and data over 64 bytes	Send PID NAK to the host Rx_error = 1	
Top Level	Receive PID OUT and data with 64 bytes	tb_usb_receive_out_64	2.2	Reading PID OUT and 64 bytes data	Send PID ACK to the host Rx_error = 0	
Top Level	Receive PID IN and send data to the host	tb_usb_receive_in_normal	2.2	Reading PID IN Receiving PID ACK after sending data	Send 4 bytes data to the host	
Top Level	Receive PID IN and send 64 bytes data to the host	tb_usb_receive_in_normal_64_bytes	2.2	Reading PID IN Receiving PID ACK after sending data	Send 64 bytes data to the host	
Top Level	Receive PID IN and send PID NAK to the host	tb_usb_receive_in_nak	2.2	Reading PID IN	Send PID NAK to the host	

AHB-Lite	Power_On_Reset	tb_ahb_reset	6	Reset	Reset all signals	
AHB-Lite	Data Buffer Register(Write)	tb_ahb_w_databuffer	6	Set hwdata and write to address **0x0** with all **hsize** considered (1,2,4 bytes)	Outputs tx_data correctly and return with no error	
AHB Slave	Data Buffer Register(Read)	tb_ahb_r_databuffer	6	Set rx_data and read from address **0x0** with all **hsize** considered (1,2,4 bytes)	Outputs rx_data correctly and return with no error 1 byte: HR Data = {24'b0, RX Data} 2 byte: HR Data = {16'b0, 2nd RX Data, 1st RX Data} 4 byte: HR Data = {4th, 3th, 2nd, 1st} (RX Data)	
AHB-Lite	Status Reg	tb_ahb_r_status	6	Read status reg from address **0x4** for rx_data_ready, rx_packet, rx_transfer_active and tx_transfer_active with all **hsize** considered (1,2,4 bytes)	New Data: HR Data = {22'b0, status[9:1], 1'b1} IN: HR Data = {22'b0, status[9:2], 1'b1, status[0]} OUT: HR Data = {22'b0, status[9:3],	

					1’b1, status[1:0]} ACK: HR Data = {22’b0, status[9:4], 1’b1, status[2:0]} NAK: HR Data = {22’b0, status[9:5], 1’b1, status[3:0]} Receive: HR Data = {22’b0, status[9], 1’b1, status[7:0]} Send: HR Data = {22’b0,1’b1, status[8:0]}	
AHB_Lite	Error Reg	Tb_ahb_error	6	Read error reg from address **0x6** for tx_error high and rx_error high with all **hsize** considered (1,2,4 bytes)	bit-0 high→ Error happened during reception of a packet bit-8 high→ Error happened during packet sending attempt	
AHB-Lite	Read Buffer Occupancy	tb_ahb_r_buffer	6	Buffer Occupancy, haddr, hsize Set buffer occupancy to a number < 64 and read from address	Hrdata with correct buffer occupancy and without errors	

				0x8 with all **hsize** considered (1,2,4 bytes)		
AHB-Lite	TX Control Reg	tb_ahb_rw_txcontrol	6	Set different hwdata between 0 to 4 and write and read to address **0xC** with all **hsize** considered (1,2,4 bytes)	Output the correct hrdata without errors 1: Send data packet 2: ACK 3: NAK 4: STALL	
AHB-Lite	Flush	tb_ahb_rw_flush	6	Flush high write and read to address **0xD** with all **hsize** considered (1,2,4 bytes)	Data buffer should be flushed after receiving flush, same as clear should output the correct hrdata without errors	
AHB-Lite	Invalid Address	tb_ahb_invalid	6	Write to invalid address Set haddr to 0xE	HRESP = 1	
AHB-Lite	Error in writing	tb_error_write	6	haddr = 0x4,5,6,7,8 Write to read only address	HRESP = 1	
Tx	One byte packet	tb_tx_one_packet	5	Send packet with one packet data	TX transmit DATA packet with correct data.	
Tx	Max byte packet	tb_tx_64_packet	5	Send packet with 64 packets data	TX transmit DATA packet	

					with correct data.	
Tx	Send STALL	tb_tx_send_stall	5	Send packet with STALL pid value	TX transmit STALL packet and d_plus signal is high	
Tx	Send ACK	tb_tx_send_ack	5	Send packet with ACK pid value	TX transmit ACK packet and d_plus signal is high	
Tx	Send NAK	tb_tx_send_nak	5	end packet with NACK pid value	TX transmit NACK packet and d_plus signal is high	
Tx	Tx Error	tb_tx_error	5	Send the tx packet while buffer occupancy is 0	Tx should respond with an error	
Buffer	Power On Reset	tb_buffer_rst	3	Reset the buffer	TX, RX and buffer occupancy should be 0	
Buffer	RX Read One Byte	tb_rx_1b	3	Store one byte into buffer from RX	Buffer occupancy increased, buffer has the value.	
Buffer	RX Read Max Byte	tb_rx_mb	3	Store max byte into buffer from RX	Buffer occupancy increased, buffer has the value.	

Buffer	TX Read 1 Byte	tb_tx_1b	3	Get one byte from buffer to TX	Buffer occupancy decreased, buffer don't have the value.	
Buffer	TX Read 64 Byte	tb_tx_mb	3	Get max byte from buffer to TX	Buffer occupancy decreased, buffer don't have the value.	
Buffer	Write One byte RX to AHB	tb_rx_ahb_1b	3	Store one byte to AHB	Correct output of rx_data	
Buffer	Write Max byte RX to AHB	tb_rx_ahb_mb	3	Store max byte to AHB	Correct output logic rx_data	
Buffer	Write One byte TX to USB	tb_tx_usb_1b	3	Get one byte data from AHB to buffer	Buffer occupancy increased, buffer has the value.	
Buffer	Write Max byte TX to USB	tb_tx_usb_mb	3	Get max byte data from AHB to buffer	Buffer occupancy increased, buffer has the value.	
Buffer	Data Flush	tb_flush	3	Flush the buffer	Data buffer should be cleared	
Rx	Receive PID IN or OUT with valid address and endpoint number	tb_rx_in_out_valid	4	Send PID IN or OUT, valid address, and valid endpoint number	Rx_packet = IN or OUT R_error = 0 NAK = 0 Rx_data_ready = 0	

Rx	Receive PID IN or OUT with invalid address or endpoint number	tb_rx_in_out_invalid	4	Send PID IN or OUT, invalid address or invalid endpoint number	Rx_packet = IN or OUT R_error = 1 NAK = 0 Rx_data_ready = 0	
Rx	Receive PID Data0 or Data1 with valid data and EOP	tb_rx_data_valid	4	Send PID DATA0 or DATA1 with valid data and valid eop	Rx_packet = DATA0 or DATA1 R_error = 0 NAK = 0 Rx_data_ready = 1	
Rx	Receive PID Data0 or Data1 having EOP during reading data	tb_rx_data_early_eop	4	Send PID DATA0 or DATA1 with valid data and invalid eop	Rx_packet = DATA0 or DATA1 R_error = 1 NAK = 0 Rx_data_ready = 1 or 0 (if error occurs at the first byte of data)	
Rx	Receive PID Data0 or Data1 reading data over 64 bytes	tb_rx_data_over64	4	Send PID DATA0 or DATA1 with data over 64 bytes	Rx_packet = DATA0 or DATA1 R_error = 0 NAK = 1 Rx_data_ready = 1	
Rx	Receive PID Data0 or Data1 reading data less	tb_rx_data_less2	4	Send PID DATA0 or DATA1 with data less than 2 bytes	Rx_packet = DATA0 or DATA1	

	than 2 bytes (no CRC)				R_error = 1 NAK = 0 Rx_data_ready = 0	
Rx	Receive PID ACK, NAK, or STALL	tb_rx_pid_ans	4	Send PID ACK, NAK, or STALL	Rx_packet = ACK, NAK, STALL R_error = 0 NAK = 0 Rx_data_ready = 0	
Rx	Receive invalid PID	tb_rx_pid_invalid	4	Send invalid PID	Rx_packet = last rx_packet R_error = 1 NAK = 0 Rx_data_ready = 0	
Rx	Receive invalid sync byte	tb_rx_syncbyte_invalid	4	Send invalid sync byte	Rx_packet = last rx_packet R_error = 1 NAK = 0 Rx_data_ready = 0	