VERIFICATION PLAN

Create by: Nguyen Van Huy

Section	Main Title	Sub Title	Description	Testname	Method	Status	Remak
1	Register test						
1.1	Read default value of register		1. Wait for reset and clock stable. 2. Read value for register: MDR, DLL, DLH, LCR, IER, FSR, TBR, RBR **Pass condition**: data matching with default value	uartlp_read_default_value_test	Direct	PASS	
1.2	Write and Read function for register		1. Wait for reset and clock stable. 2. Write and read 1's and 0's in each bit of the registe. For registerMDR, DLL, DLH, LCR, IER.. Pass condition: data matching between value write and value read	uartlp_reg_scan_test	Direct	PASS	
1.3	Write and read function for rsvd register		1. Wait for reset and clock stable. 2. Write and read Reserved region register from address 0X020-0X3FF Pass condition: data read match value 32'FFFF_FFFF	uartlp_rw_rsvd_reg_test	Direct	PASS	
1.4	Write and read function for FSR register		1. Wait for reset and clock stable. 2.Configure the register for the RX port of the UART IP to receive data. 3. Configure UART IP and UART VIP with different parity bit settings (to simulate a parity bit error). 4. CHECK parrity_error_status when UART IP complete Reception. 5. Send random data to the RX port of the UART IP. 6. CHECK parrity_error_status when UART IP complete Reception. 7. Write one to clear the FSR register to clear the parity_error_status bit. 8. CHECK parrity_error_status. **Pass condition**: The FSR register operates correctly in W1C mode	uartlp_FSR_reg_test	Direct	FAIL	
2	Transmission test						
2.4	Baud rate with 16x and 13x oversampling test	baud rate 2400 and 16x, 13x oversampling	Setting clock: Mode x16, baud rate 2400,4800,9600,19200,38400,76800,115200. Setting uart frame: No/Even/Odd parity, 1/2stop bit, 5/6/7/8 data bit. **First transfer** 1. Wait for reset and clock stable. 2. Configure mode for UART sampling in register MDR(Mode x16). 3. Configure divisor value in register DLL and register DLH (baurate (2400,4800,9600,19200,38400,76800,115200). 4. Enable baud generator at bit 5 (BGE) in register LCR. 5. Configure UART FRAME in register LCR. 6. Write data to buffer in register TBR **Second transfer** 7. Configure mode for UART sampling in register MDR(Mode x13) 8. Configure divisor value in register DLL and register DLH (baurate (2400,4800,9600,19200,38400,76800,115200). 9. Write data to buffer in register TBR **Pass conditions**: data Uart IP mathching Uart VIP, correct Start bit, Stop bit, Parity bit from transmission for UART VIP. Data is not changed within the bit duration. **Checker:** + Uartlp_test.uartlp_env.uartlp_sb: compere_data_check, parity_bit_check, stop_bit_check, start_bit_check + Uartlp_test.env.uart_agent.uart_mon: baud_rate_check + Uartlp_test: detect_difference_config	uartlp_trans_baudrate2400_test	Direct, Checker	PASS	
		baud rate 4800 and 16x, 13x oversampling		uartlp_trans_baudrate4800_test	Direct, Checker	PASS	
		baud rate 9600 and 16x ,13x oversampling		uartlp_trans_baudrate9600_test	Direct, Checker	PASS	
		baud rate 19200 and 16x, 13x oversampling		uartlp_trans_baudrate19200_test	Direct, Checker	PASS	
		baud rate 38400 and 16x, 13x oversampling		uartlp_trans_baudrate38400_test	Direct, Checker	PASS	
		baud rate 76800 and 16x, 13x oversampling		uartlp_trans_baudrate76800_test	Direct, Checker	PASS	
		baud rate 115200 and 16x, 13x oversampling		uartlp_trans_baudrate115200_test	Direct, Checker	PASS	
2.5	Uart IP basic function	1 stop bit, No parity bit, 5 data bit	Setting clock: Mode x16, baud rate 9600. Setting uart frame: No/Even/Odd parity, 1/2stop bit, 5/6/7/8 data bit. **First transfer** 1. Wait for reset and clock stable. 2. Configure mode for UART sampling in register MDR. 3. Configure divisor value in register DLL and register DLH. 4. Enable baud generator at bit 5 (BGE) in register LCR. 5. Configure UART FRAME in register LCR. 6. Write data to buffer in register TBR (data causes the parity bit to be set to 1 or 0) **Second transfer** (checking No parity bit Not apply) 7. Write data to buffer in register TBR (data causes the parity bit to be set to 0 or 1) **Pass conditions:** data Uart IP mathching Uart VIP, correct Start bit, Stop bit, Parity bit from transmission for UART VIP. **Checker:** + Uartlp_test.uartlp_env.uartlp_sb: compere_data_check, parity_bit_check, stop_bit_check, start_bit_check + Uartlp_test: detect_difference_config	uartlp_trans_1stopbit_noparitybit_5databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, No parity bit, 6 data bit		uartlp_trans_1stopbit_noparitybit_6databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, No parity bit, 7 data bit		uartlp_trans_1stopbit_noparitybit_7databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, No parity bit, 8 data bit		uartlp_trans_1stopbit_noparitybit_8databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, Even parity bit, 5 data bit		uartlp_trans_1stopbit_evenparitybit_5databit_test	Direct, constrain random, checker	FAIL	
		1 stop bit, Even parity bit, 6 data bit		uartlp_trans_1stopbit_evenparitybit_6databit_test	Direct, constrain random, checker	FAIL	
		1 stop bit, Even parity bit, 7 data bit		uartlp_trans_1stopbit_evenparitybit_7databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, Even parity bit, 8 data bit		uartlp_trans_1stopbit_evenparitybit_8databit_test	Direct, constrain random, checker	FAIL	
		1 stop bit, Odd parity bit, 5 data bit		uartlp_trans_1stopbit_oddparitybit_5databit_test	Direct, constrain random, checker	FAIL	
		1 stop bit, Odd parity bit, 6 data bit		uartlp_trans_1stopbit_oddparitybit_6databit_test	Direct, constrain random, checker	FAIL	
		1 stop bit, Odd parity bit, 7 data bit		uartlp_trans_1stopbit_oddparitybit_7databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, Even parity bit, 8 data bit		uartlp_trans_1stopbit_oddparitybit_8databit_test	Direct, constrain random, checker	FAIL	
		2 stop bit, No parity bit, 5 data bit		uartlp_trans_2stopbit_noparitybit_5databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, No parity bit, 6 data bit		uartlp_trans_2stopbit_noparitybit_6databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, No parity bit, 7 data bit		uartlp_trans_2stopbit_noparitybit_7databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, No parity bit, 8 data bit		uartlp_trans_2stopbit_noparitybit_8databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, Even parity bit, 5 data bit		uartlp_trans_2stopbit_evenparitybit_5databit_test	Direct, constrain random, checker	FAIL	
		2 stop bit, Even parity bit, 6 data bit		uartlp_trans_2stopbit_evenparitybit_6databit_test	Direct, constrain random, checker	FAIL	
		2 stop bit, Even parity bit, 7 data bit		uartlp_trans_2stopbit_evenparitybit_7databit_test	Direct, constrain random, checker	FAIL	
		2 stop bit, Even parity bit, 8 data bit		uartlp_trans_2stopbit_evenparitybit_8databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, Odd parity bit, 5 data bit		uartlp_trans_2stopbit_oddparitybit_5databit_test	Direct, constrain random, checker	FAIL	
		2 stop bit, Odd parity bit, 6 data bit		uartlp_trans_2stopbit_oddparitybit_6databit_test	Direct, constrain random, checker	FAIL	
		2 stop bit, Odd parity bit, 7 data bit		uartlp_trans_2stopbit_oddparitybit_7databit_test	Direct, constrain random, checker	FAIL	
		2 stop bit, Even parity bit, 8 data bit		uartlp_trans_2stopbit_oddparitybit_8databit_test	Direct, constrain random, checker	PASS	
	FIFO test	FIFO full	1. Wait for reset and clock stable. 2. Configure mode for UART sampling in register MDR. 3. Configure divisor value in register DLL and register DLH. 4. Enable baud generator at bit 5 (BGE) in register LCR. 5. Configure UART FRAME in register LCR. 6. Write **18 data** to buffer in register TBR. UART IP will transmit 17 data. If UART VIP receives exactly 17 data, the test passes. If it receives more than 17 data, the test fails	uartlp_trans_2stopbit_oddparitybit_8databit_test	Direct	FAIL	
3	Reception test						
3.4	Baud rate with 16x and 13x oversampling test	baud rate 2400 and 16x oversampling	Setting clock: Mode x16, baud rate 2400,4800,9600,19200,38400,76800,115200. Setting uart frame: No/Even/Odd parity, 1/2stop bit, 5/6/7/8 data bit. **First transfer** 1. Wait for reset and clock stable. 2. Configure mode for UART sampling in register MDR(Mode x16). 3. Configure divisor value in register DLL and register DLH (baurate (2400,4800,9600,19200,38400,76800,115200). 4. Enable baud generator at bit 5 (BGE) in register LCR. 5. Configure UART FRAME in register LCR. 6. Send random data to the RX port of the UART IP (data causes the parity bit to be set to 1 or 0). 7. Read data from the RBR register. **Second transfer** 8. Configure mode for UART sampling in register MDR(Mode x13). 9. Configure divisor value in register DLL and register DLH (baurate (2400,4800,9600,19200,38400,76800,115200). 10. Send random data to the RX port of the UART IP (data causes the parity bit to be set to 0 or 1). 11. Read data from the RBR register **Pass conditions:** data Uart IP mathching Uart VIP. **Checker:** + Uartlp_test.uartlp_env.uartlp_sb: compere_data_check. + Uartlp_test: detect_difference_config	uartlp_recep_baudrate2400_test	Direct, Checker	PASS	
		baud rate 4800 and 13x oversampling		uartlp_recep_baudrate4800_test	Direct, Checker	PASS	
		baud rate 9600 and 16x oversampling		uartlp_recep_baudrate9600_test	Direct, Checker	PASS	
		baud rate 19200 and 13x oversampling		uartlp_recep_baudrate19200_test	Direct, Checker	PASS	
		baud rate 38400 and 16x oversampling		uartlp_recep_baudrate38400_test	Direct, Checker	PASS	
		baud rate 76800 and 13x oversampling		uartlp_recep_baudrate76800_test	Direct, Checker	PASS	
2.5	Uart IP basic function	1 stop bit, No parity bit, 5 data bit	Setting clock: Mode x16, baud rate 9600. Setting uart frame: No/Even/Odd parity, 1/2stop bit, 5/6/7/8 data bit. **First transfer** 1. Wait for reset and clock stable. 2. Configure mode for UART sampling in register MDR. 3. Configure divisor value in register DLL and register DLH. 4. Enable baud generator at bit 5 (BGE) in register LCR. 5. Configure UART FRAME in register LCR. 6. Send random data to the RX port of the UART IP. 7. Read data from the RBR register **Second transfer** (checking No parity bit Not apply) 8. Send random data to the RX port of the UART IP. 9. Read data from the RBR register **Pass conditions:** data Uart IP mathching Uart VIP, correct Start bit, Stop bit, Parity bit from transmission for UART VIP. **Checker:** + Uartlp_test.uartlp_env.uartlp_sb: compere_data_check, parity_bit_check, stop_bit_check, start_bit_check + Uartlp_test: detect_difference_config	uartlp_recep_1stopbit_noparitybit_5databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, No parity bit, 6 data bit		uartlp_recep_1stopbit_noparitybit_6databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, No parity bit, 7 data bit		uartlp_recep_1stopbit_noparitybit_7databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, No parity bit, 8 data bit		uartlp_recep_1stopbit_noparitybit_8databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, Even parity bit, 5 data bit		uartlp_recep_1stopbit_evenparitybit_5databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, Even parity bit, 6 data bit		uartlp_recep_1stopbit_evenparitybit_6databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, Even parity bit, 7 data bit		uartlp_recep_1stopbit_evenparitybit_7databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, Even parity bit, 8 data bit		uartlp_recep_1stopbit_evenparitybit_8databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, Odd parity bit, 5 data bit		uartlp_recep_1stopbit_oddparitybit_5databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, Odd parity bit, 6 data bit		uartlp_recep_1stopbit_oddparitybit_6databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, Odd parity bit, 7 data bit		uartlp_recep_1stopbit_oddparitybit_7databit_test	Direct, constrain random, checker	PASS	
		1 stop bit, Odd parity bit, 8 data bit		uartlp_recep_1stopbit_oddparitybit_8databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, No parity bit, 5 data bit		uartlp_recep_2stopbit_noparitybit_5databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, No parity bit, 6 data bit		uartlp_recep_2stopbit_noparitybit_6databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, No parity bit, 7 data bit		uartlp_recep_2stopbit_noparitybit_7databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, No parity bit, 8 data bit		uartlp_recep_2stopbit_noparitybit_8databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, Even parity bit, 5 data bit		uartlp_recep_2stopbit_evenparitybit_5databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, Even parity bit, 6 data bit		uartlp_recep_2stopbit_evenparitybit_6databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, Even parity bit, 7 data bit		uartlp_recep_2stopbit_evenparitybit_7databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, Even parity bit, 8 data bit		uartlp_recep_2stopbit_evenparitybit_8databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, Odd parity bit, 5 data bit		uartlp_recep_2stopbit_oddparitybit_5databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, Odd parity bit, 6 data bit		uartlp_recep_2stopbit_oddparitybit_6databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, Odd parity bit, 7 data bit		uartlp_recep_2stopbit_oddparitybit_7databit_test	Direct, constrain random, checker	PASS	
		2 stop bit, Even parity bit, 8 data bit		uartlp_recep_2stopbit_oddparitybit_8databit_test	Direct, constrain random, checker	PASS	

4	Transmission and Reception test						
3.4	Baud rate with 16x and 13x oversampling test	baud rate 2400 and 16x oversampling	Setting clock: Mode x16, baud rate 2400,4800,9600,19200,38400,76800,115200. Setting uart frame: No/Even/Odd parity, 1/2stop bit, 5/6/7/8 data bit. **First transfer** 1. Wait for reset and clock stable. 2. Configure mode for UART sampling in register MDR(Mode x16). 3. Configure divisor value in register DLL and register DLH (baurate (2400,4800,9600,19200,38400,76800,115200). 4. Enable baud generator at bit 5 (BGE) in register LCR. 5. Configure UART FRAME in register LCR. 6. Write data to buffer in register TBR 7. Send random data to the RX port of the UART IP. 8. Read data from the RBR register **Second transfer** 9. Configure mode for UART sampling in register MDR(Mode x13) 10. Configure divisor value in register DLL and register DLH (baurate (2400,4800,9600,19200,38400,76800,115200). 11. Write data to buffer in register TBR 12. Send random data to the RX port of the UART IP. 13. Read data from the RBR register **Pass conditions**: data Uart IP mathching Uart VIP, correct Start bit, Stop bit, Parity bit from transmission for UART VIP. Data is not changed within the bit duration. **Checker:** + Uartip_test.uartip_env.uartip_sb: compere_data_check, parity_bit_check, stop_bit_check, start_bit_check. + Uartip_test.env.uart_agent.uart_mon: baud_rate_check. + Uartip_test: detect_difference_config.	uartip_trans_and_recep_baudrate2400_test	Direct	PASS	
		baud rate 4800 and 13x oversampling		uartip_trans_and_recep_baudrate4800_test	Direct	PASS	
		baud rate 9600 and 16x oversampling		uartip_trans_and_recep_baudrate9600_test	Direct	PASS	
		baud rate 19200 and 13x oversampling		uartip_trans_and_recep_baudrate19200_test	Direct	PASS	
		baud rate 38400 and 16x oversampling		uartip_trans_and_recep_baudrate38400_test	Direct	PASS	
		baud rate 76800 and 13x oversampling		uartip_trans_and_recep_baudrate76800_test	Direct	PASS	
		baud rate 115200 and 16x oversampling		uartip_trans_fifo_full_test	Direct	PASS	
2.5	Uart IP basic function	1 stop bit, No parity bit, 5 data bit	Setting clock: Mode x16, baud rate 9600. Setting uart frame: No/Even/Odd parity, 1/2stop bit, 5/6/7/8 data bit. **First transfer** 1. Wait for reset and clock stable. 2. Configure mode for UART sampling in register MDR. 3. Configure divisor value in register DLL and register DLH. 4. Enable baud generator at bit 5 (BGE) in register LCR. 5. Configure UART FRAME in register LCR. 6. Send random data to the RX port of the UART IP. 7. Read data from the RBR register 8. Send random data to the RX port of the UART IP. 9. Read data from the RBR register **Second transfer** (checking No parity bit Not apply) 10. Send random data to the RX port of the UART IP. 11. Read data from the RBR register 12. Send random data to the RX port of the UART IP. 13. Read data from the RBR register **Pass conditions:** data Uart IP mathching Uart VIP, correct Start bit, Stop bit, Parity bit from transmission for UART VIP. **Checker:** + Uartip_test.uartip_env.uartip_sb: compere_data_check, parity_bit_check, stop_bit_check, start_bit_check + Uartip_test: detect_difference_config	uartip_trans_and_recep_1stopbit_noparitybit_5databit_test	Direct	PASS	
		1 stop bit, No parity bit, 6 data bit		uartip_trans_and_recep_1stopbit_noparitybit_6databit_test	Direct	PASS	
		1 stop bit, No parity bit, 7 data bit		uartip_trans_and_recep_1stopbit_noparitybit_7databit_test	Direct	PASS	
		1 stop bit, No parity bit, 8 data bit		uartip_trans_and_recep_1stopbit_noparitybit_8databit_test	Direct	PASS	
		1 stop bit, Even parity bit, 5 data bit		uartip_trans_and_recep_1stopbit_evenparitybit_5databit_test	Direct	FAIL	
		1 stop bit, Even parity bit, 6 data bit		uartip_trans_and_recep_1stopbit_evenparitybit_6databit_test	Direct	FAIL	
		1 stop bit, Even parity bit, 7 data bit		uartip_trans_and_recep_1stopbit_evenparitybit_7databit_test	Direct	PASS	
		1 stop bit, Even parity bit, 8 data bit		uartip_trans_and_recep_1stopbit_evenparitybit_8databit_test	Direct	FAIL	
		1 stop bit, Odd parity bit, 5 data bit		uartip_trans_and_recep_1stopbit_oddparitybit_5databit_test	Direct	FAIL	
		1 stop bit, Odd parity bit, 6 data bit		uartip_trans_and_recep_1stopbit_oddparitybit_6databit_test	Direct	FAIL	
		1 stop bit, Odd parity bit, 7 data bit		uartip_trans_and_recep_1stopbit_oddparitybit_7databit_test	Direct	PASS	
		1 stop bit, Odd parity bit, 8 data bit		uartip_trans_and_recep_1stopbit_oddparitybit_8databit_test	Direct	FAIL	
		2 stop bit, No parity bit, 5 data bit		uartip_trans_and_recep_2stopbit_noparitybit_5databit_test	Direct	PASS	
		2 stop bit, No parity bit, 6 data bit		uartip_trans_and_recep_2stopbit_noparitybit_6databit_test	Direct	PASS	
		2 stop bit, No parity bit, 7 data bit		uartip_trans_and_recep_2stopbit_noparitybit_7databit_test	Direct	PASS	
		2 stop bit, No parity bit, 8 data bit		uartip_trans_and_recep_2stopbit_noparitybit_8databit_test	Direct	PASS	
		2 stop bit, Even parity bit, 5 data bit		uartip_trans_and_recep_2stopbit_evenparitybit_5databit_test	Direct	FAIL	
		2 stop bit, Even parity bit, 6 data bit		uartip_trans_and_recep_2stopbit_evenparitybit_6databit_test	Direct	FAIL	
		2 stop bit, Even parity bit, 7 data bit		uartip_trans_and_recep_2stopbit_evenparitybit_7databit_test	Direct	FAIL	
		2 stop bit, Even parity bit, 8 data bit		uartip_trans_and_recep_2stopbit_evenparitybit_8databit_test	Direct	PASS	
		2 stop bit, Odd parity bit, 5 data bit		uartip_trans_and_recep_2stopbit_oddparitybit_5databit_test	Direct	FAIL	
		2 stop bit, Odd parity bit, 6 data bit		uartip_trans_and_recep_2stopbit_oddparitybit_6databit_test	Direct	FAIL	
		2 stop bit, Odd parity bit, 7 data bit		uartip_trans_and_recep_2stopbit_oddparitybit_7databit_test	Direct	FAIL	
		2 stop bit, Even parity bit, 8 data bit		uartip_trans_and_recep_2stopbit_oddparitybit_8databit_test	Direct	PASS	
5	Error handling test						
5.1	Access reserved address		1. Wait for reset and clock stable. 2. Write and read Reserved region register from address 0X020-0X3FF Pass condition: HRESP signal equa to 1		Direct	PASS	
6	Interrupt and FIFO status test						
6.1	Enable Parity bit error	Even parity error	1.Configure the register for the RX port of the UART IP to receive data. 2.Configure the IER register to Enable parity error Hardware interrupt. 3.**No parity error** 3.1. **CHECK** the interrupt and parity_error_status before UART IP receives data 3.2. Send random data to the RX port of the UART IP (**data causes the parity bit to be set to 1 and 0**). 3.3. **CHECK** the interrupt and parrity_error_status when UART IP complete Reception. 3.4. Read data from the RBR register to clear data in the RX FIFO. 4.**Parity error** 4.1 Configure UART IP and UART VIP with different parity bit settings (to simulate a parity bit error). 4.2 **CHECK** the interrupt and parity_error_status before UART IP receives data 4.3 Send random data to the RX port of the UART IP (**data causes the parity bit to be set to 1 and 0**). 4.4 **CHECK** the interrupt and parrity_error_status when UART IP complete Reception. 4.5 **Write one to clear** the FSR register to clear the parity_error_status bit. 4.6 **CHECK** parrity_error_status. 4.7 Read data from the RBR register to clear data in the RX FIFO. 5. **Repeat one more time** with data that **causes the parity bit to be set to 0 and 1.** **Pass conditon:** - The interrupt signal is 1 and parrity_error_status is 1 when Parity error. - The interrupt signal is 0 and parrity_error_status is 0 when No parity error. **Checker:** + Uartip_test.uartip_env.uartip_sb: compere_data_check. + Uartip_test: FSR_bit_check, interrupt_signal_check.	uartip_interrupt_enable_evenparity_test	Direct, Checker	FAIL	- The Interrupt and Parity_Error_Status of the FSR are not working and remain at their default values. The Interrupt and Parity_Error_Status are always 0
		Odd parity error		uartip_interrupt_enable_oddparity_test	Direct, Checker	FAIL	- The Interrupt and Parity_Error_Status of the FSR are not working and remain at their default values. The Interrupt and Parity_Error_Status are always 0
6.2	Enable RX FIFO test	FIFO empty	1.Configure the register for the RX port of the UART IP to receive data. 2. Configure the IER register to enable the RX FIFO empty hardware interup. 3. **CHECK** the **interrupt** and **rx_empty_status** bit when the RX FIFO is **empty**. 4.Send **one** random data to the RX port of the UART IP to make the RX FIFO contain data. 5. **CHECK** the **interrupt** and **rx_empty_status** bit when the RX FIFO **contain data**. 6. Read data from the RBR register to make the RX FIFO is empty. 7. **CHECK** the **interrupt** and **rx_empty_status** bit when the RX FIFO is **empty**. **Pass conditon:** - The **interrupt** signal is 1 and **rx_empty_status** is 0 when the FIFO is **empty**. - The **interrupt** signal is 0 and **rx_empty_status** is 1 when the FIFO **contains data**. **Checker:** + Uartip_test.uartip_env.uartip_sb: compere_data_check. + Uartip_test: FSR_bit_check, interrupt_signal_check.	uartip_interrupt_enable_rxfifofull_test	Direct, Checker	FAIL	- The interrupt functions correctly, but the rx_empty_status of the FSR is functioning incorrectly. The rx_empty_status bit has its function inverted: it is 1 when the FIFO is empty and 0 when the FIFO contains data.
		FIFO full	1.Configure the register for the RX port of the UART IP to receive data. 2. Configure the IER register to enable the RX FIFO full hardware interup. 3. **CHECK** the **interrupt** and **rx_full_status** bit when the RX FIFO has **no full**. 4.Send **16** random data to the RX port of the UART IP to make the RX FIFO is **full**. 5. **CHECK** the **interrupt** and **rx_full_status** bit when the RX FIFO is **full**. 6. Read data from the RBR register to make the RX FIFO has **no full**. 7. **CHECK** the **interrupt** and **rx_full_status** bit when the RX FIFO has **no full**. **Pass conditon:** - The **interrupt** signal is 1 and **rx_full_status** is 1 when the FIFO has **no data**. - The **interrupt** signal is 0 and **rx_full_status** is 0 when the FIFO is **full**. **Checker:** + Uartip_test.uartip_env.uartip_sb: compere_data_check. + Uartip_test: FSR_bit_check, interrupt_signal_check.	uartip_interrupt_enable_rxfifofull_test	Direct, Checker	FAIL	- The interrupt and the RX_Full_Status of the FSR function correctly, but the RX_Empty_Status has changed even though this bit was not enabled.
6.3	Enable TX FIFO test	FIFO empty	1.Configure the register for the TX port of the UART IP to transmit data. 2. Configure the IER register to enable the TX FIFO empty hardware interup. 3. **CHECK** the **interrupt** and **tx_empty_status** bit when the TX FIFO is **empty**. 4.Write data to buffer in register TBR of the UART IP to make the TX FIFO contain data. 5. **CHECK** the **interrupt** and tx_**empty_status** bit when the RX FIFO **contain data**. 6. Wait for UART IP to complete transmission until TX FIFO is **empty.** 7. **CHECK** the **interrupt** and tx_**empty_status** bit when the RX FIFO is **empty**. **Pass conditon:** - The **interrupt** signal is 1 and **tx_empty_status** is 0 when the FIFO is **empty**. - The **interrupt** signal is 0 and **tx_empty_status** is 1 when the FIFO **contains data**. **Checker:** + Uartip_test.uartip_env.uartip_sb: compere_data_check. + Uartip_test: FSR_bit_check, interrupt_signal_check.	uartip_interrupt_enable_rxfifofull_test	Direct, Checker	FAIL	The Interrupt and TX_Full_Status of the FSR are not working and remain at their default values. The Interrupt is always 1, and TX_Full_Status is always 1.
		FIFO full	1.Configure the register for the TX port of the UART IP to transmit data. 2. Configure the IER register to enable the TX FIFO full hardware interup. 3. **CHECK** the **interrupt** and **tx_full_status** bit when the TX FIFO has **no full**. 4.Write **17** data to buffer in register TBR of the UART IP to make the TX FIFO is **full**. 5. **CHECK** the **interrupt** and **tx_full_status** bit when the RX FIFO is **full**. 6. Wait for UART IP to complete transmission until TX FIFO has **no full**. 7. **CHECK** the **interrupt** and **tx_full_status** bit when the RX FIFO has **no full**. **Pass condition:** - The **interrupt** signal is 1 and **rx_full_status** is 1 when the FIFO has **no data**. - The **interrupt** signal is 0 and **rx_full_status** is 0 when the FIFO is **full**. **Checker:** + Uartip_test.uartip_env.uartip_sb: compere_data_check. + Uartip_test: FSR_bit_check, interrupt_signal_check.	uartip_interrupt_enable_rxfifofull_test	Direct, Checker	FAIL	-The interrupt functions correctly, but thetx_full_status of the FSR is not working. The tx_full_status bit is 0 when the interrupt is 1