Milestone	Done	Test Name	Feature	Test Description	Test type	Test Stimulus	Reference	Status	Comments	Results	Document
Transmission Tests (RS232)	☑	single_tx_test	**Transmission** operation by all **randomized** configuration settings	Run a transmission with randomized configuration settings	C-test	1. Apply reset for some clock cycles 2. Write 1 to the LCR[7] i.e. DLAB bit to access DLL/DLH CSRs 3. Write to DLL,DLH to set the divisor for the baudrate 4. Write 0 to the LCR[7] i.e. DLAB bit 5. Randomize the configuration settings (parity,stop bit, character length, break cntrl bit) and write the LCR register 6. Program the modem control register (optional in this case but necessary if fifos are enabled) 7. Write a randomized character (to be transmitted) in THR register 8. Monitor the *"sout"* pin via VIP monitor according to the configuration settings and write the monitored data to analysis_export port of scoreboard 9. Update mailbox with the monitored character in the scoreboard 10. Wait in C-test for the scoreboard to update mailbox with the monitored payload 11. Compare the expected data written to THR and the received data from scoreboard via mailbox **Repeat the test for sending more character bytes with other randomized cfg settings**	DW_apb_uart_ databook section:: 1.2.1 6.1 5.1.8 5.1.2 5.1.4 5.1.3 5.1.7	**PASSED**	* None of the interrupt enable * No FIFO *also add repeatitions		
	☑	fifo_enabled_tx_test	**Transmission** operation with **enabled FIFO**	Run some transmissions on IP by enabling TX-FIFO and with the randomized configuration	C-test	**Configure the IP** (as Step 1-6 in single_tx_test) 7. Write 1 to the FCR[0] (FCROE) for enabling FIFO 8. Halt the transmission by setteing HTX bit of HTX register 9. Write multiple Randomized characters (to be transmitted) in THR register and fill the FIFO 10. Clear the HTX bit of HTX register to un-halt transmission and transmission will start 11. Monitor the "sout" pin via VIP monitor according to the configuration settings and write the monotored data to export port of scoreboard 12. Wait in C-test for the scoreboard to update mailbox with the monitored payload 13. Compare the expected data written to THR and the received data from scoreboard via mailbox **Repeat the test multiple times**	DW_apb_uart_ databook section:: 1.2.1 6.1 5.1.8 5.1.2 5.1.4 5.1.3 5.1.6 5.1.7	**PASSED**	* FIFO enabled for multiple transmissions. C-test require some touchup modifications		
Receive Tests (RS232)	☑	single_rec_test	Single Receive operation by all the randomized configuration settings	Run a receive test with all the random configuration settings	C-test & sequence	**Configure the IP** (as Step 1-6 in single_tx_test) 7. Drive payload on "sin" pin via VIP driver w.r.t. the configuration settings. IP will start receiving data as soon the start-bit received at "sin" pin 8. As the driver completes driving a payload, write the transaction to the scoreboard's analysis_export port 9. Scoreboard updates mailbox with the payload drived for the c-test 10. Wait in c-test for scoreboard to update mailbox 11. Read the RBR register 12. Compare the expected payload driven at sout and the received data read from RBR **Repeat the test multiple times**	DW_apb_uart_ databook section:: 1.2.1 6.1 5.1.8 5.1.2 5.1.4 5.1.3 5.1.7	**PASSED**	* None of the interrupt enable * No FIFO		
	☑	fifo_enabled_rec_test	Receive operation with **enabled FIFO**	Run some receive tests on IP by enabling RX-FIFO and randomized configuration settings	C-test & sequence	**Configure the IP** (as Step 1-6 in single_tx_test) 7. Write to FCR for enabling FIFO and setting threshold level 7.1. Write 1 to the FCR[0] (FCROE) for enabling FIFO 7.2. Write at FCR[7:6] (RT) bit for RX FIFO trigger level 8. Run a sequence multiple times to Drive random payloads on "sin" pin via VIP driver w.r.t. the configuration settings. IP will start receiving data as soon the start-bit received at "sin" pin 9. As the driver completes driving a payload, write the transaction to the scoreboard's analysis_export port 10. Scoreboard updates mailbox with the payload drived for the c-test 11. Wait in c-test for scoreboard to update mailbox 12. Read the RBR register 13. Compare the expected payload driven at sout and the received data read from RBR	DW_apb_uart_ databook section:: 1.2.1 6.1 5.1.8 5.1.2 5.1.4 5.1.3 5.1.6 5.1.7	**PASSED**	* FIFO enabled for regressive testing. C-test require some touchup modifications		
Rec Errors Test	☑	RecLineErr	To test the receive line error (overrun error, framing error, parity error, break error) set/clear logic	**OE:** Drive the sout for receive operation on IP when the received FIFO is already full (IF FIFO is enable) or RBR already having unread data. Read the corresponding error bit i.e. OE bit of LSR. **FE:** Run receive sequence with intentially corrupted stop bit and read the corresponding error bit i.e. FE bit of LSR. **PE:** Intentially corrupt the parity bit and run receive sequence on IP and read the corresponding error bit i.e. PE bit of LSR. **BE:** Intentially send break sequences on sout pin of IP and read error from the corresponding error bit i.e. BI bit of LSR. **The errro bit should be cleared by IP upon reading LSR**	C-test & sequence	**Configure the IP** (as Step 1-6 in single_tx_test) **and step-7 (enabling of FIFO and setting of threshold level) is optional** 8. Run any of the following erroneous sequence -Drive payload on "sin" pin via VIP driver w.r.t. the configuration settings until the threshold level crossed (for OE) -Drive the stop bit corrupted payload on "sin" pin via VIP driver w.r.t. the configuration settings (for FE) -Drive the parity bit corrupted payload on "sin" pin via VIP driver w.r.t. the configuration settings (for PE) -Drive the break sequence on "sin" pin (i.e. all zeroes for whole frame) via VIP driver according to the configuration settings (for BE) IP will start receiving data as soon the start-bit received at "sin" pin. 9. As the driver completes driving a payload, write the transaction to the scoreboard's analysis_export port 10. Scoreboard updates mailbox with the payload drived for the c-test 11. Wait in c-test for scoreboard to update mailbox 12. Read the RBR register 13. Compare the expected payload driven at sout and the received data read from RBR 14. Read the LSR register and observe the corresponding bit (should be set to 1) -OE bit (LSR[1]) for overrun error -FE bit (LSR[3]) for frame error -PE bit (LSR[2]) for parity error -BI bit (LSR[4]) for break error **Repeat the test to cover all the receive errors**	DW_apb_uart_ databook section:: 1.2.1 6.1 5.1.8 5.1.2 5.1.4 5.1.3 5.1.6 5.1.7 5.1.10	merged with receive line interrupt tests	The test is pretty much same as the receive line interrupt. So it is merged with receive line inerrupt test		

Interrupts Tests	☑	rx_p_err_test / rx_f_err_test	Receiver line status Interrupt Set/clear test	Intentially run erroneous sequences and read the IIR register to check the respective interrupt i.e receive line interrupt in this particular case. Than clear that interrupt by LSR read, LSR also checks the interrupt source	C-test & sequence	**Configure the IP** (as Step 1-6 in single_tx_test) 7. Write to IER register to enable the required interrupts i.e. enable ELSI bit 8. Run any erroneous sequences-(PE / FE / OE / BI) 9. Interrupt should occur 10. Handler of the interrupt in C-test 10.1. Reading IIR[3:0] (IID) register to find the respective interrupt. IID would be 4'b0110 indicating the receive line error 10.2. Read the LSR register to know the cause of interrupt. Handle the interrupt in the respective manner 10.3. Interrupt should gets cleared by reading LSR 10.4. Update the respective mailbox entry for parity error and frame error 11. Compare the expected data, parity error, frame error (drived in VIP) and received data, parity error, frame error (via APB interface using C-test)	DW_apb_uart_ databook section:: 1.2.1 6.1 5.1.8 5.1.2 5.1.4 5.1.3 5.1.5 5.1.5 5.1.6 5.1.7 2.9	PASSED	BI and OE is remaining		
	☑	txrx_intr_test	Receiver data availbale and transmit hoding register empty Interrupt Set/clear test	Receive data avilable (FIFO disable) or RCVR FIFO trigger level reached. Transmit holding register empty (FIFO disable) or TX FIFO is empty	C-test & sequence	**Configure the IP** (as Step 1-6 in single_tx_test) 7. Write to IER register to enable the required interrupts, i.e enable ETBEI, ERBFI bits 8. Interrupt would occur as the transmit holding register is empty 9. Reading IIR would down this interrupt 10. Drive some rx sequences without reading the RBR untill the FIFO trigger level reached (if FIFO enabled) 11. Interrupt should occur as the receive data available 12. Handler of the interrupt in C-test 12.1. Reading IIR[3:0] (IID) register to find the respective interrupt. IID would be 4'b0100 indicating the receive data available 12.2. Interrupt should gets cleared by one RBR read (FIFO disable) or multiple read untill FIFO drops below trigger level (FIFO enable) 13. compare the drived and received payloads	DW_apb_uart_ databook section:: 1.2.1 6.1 5.1.8 5.1.2 5.1.4 5.1.3 5.1.5 5.1.5 5.1.6 5.1.7 2.9	PASSED			
	☑	timout_intr_test	character timeout interrupt	No character in or out of the RCVR FIFO for four character time, while there is atleast one character in the FIFO	C-test & sequence	**Configure the IP** (as Step 1-6 in single_tx_test) 7. Write to IER register to enable the required interrupt, i.e. 8. Drive rx sequencen from VIP 8.1. Do not perform any operation for atleast for 4 character time 8.2. This would cause interrupt to occure as timeout interrupt 9. Handler of the interrupt in C-test 9.1. Reading IIR[3:0] (IID) register to find the respective interrupt. IID would be 4'b1100 indicating the timeout interrupt 9.2. Interrupt should gets cleared by one RBR read 10. compare the drived and received payload	DW_apb_uart_ databook section:: 1.2.1 6.1 5.1.8 5.1.2 5.1.4 5.1.3 5.1.5 5.1.5 5.1.6 5.1.7 2.9	PASSED			
	☑	ri_ind_intr_test	Ring Indicator port ri_n cause modem status interupt	Assertion of ri_n (input port to IP) cause modem status interrupt	C-test & sequence	Configure the IP (as Step 1-6 in single_tx_test) 7. Write to IER register to enable the required interrupt, i.e.EDSSI 8. Drive the ri_n to LOW in the modem agent. 9. Wait for modem status interrupt to occure and handle in interrupt service routine function. 10. Interrupt gets down by MSR read 11. Check the MSR value, MSR[6] (RI bit) should be HIGH-(cause of interrupt)	DW_apb_uart_ databook section:: 1.2.1 2.9 5.1 5.1.11 6.1	PASSED			
DMA Tests	☑	dma_tx_test	DMA interface during transmission	To test the DMA operations and DMA TX interface signals i.e. dma_tx_ack_n, dma_tx_req_n, dma_tx_single_n	C-test	**Configure the IP** (as Step 1-6 in single_tx_test) 6. Write to FCR for enabling FIFO and setting threshold level also called watermark level 6.1. Write 1 to the FCR[0] (FCROE) for enabling FIFO 6.2. FCR[5:4] (TET) is not programmable in our case so the threshould is set as fifo_empty bydefault 7. Write 1 to HTX register to halt the transmission 8. IP will request to write into TX-FIFO using DMA I/F signals. Monitor the signal in the respective agent and communicate to c-test via mailbox 9. Fill the TX-FIFO by writing to THR upto the imaginary burst size selected 10. Communicate via mailbox to the respective agent to drive "dma_tx_ack" to HIGH 11. dma_tx_req must down now 12. Un-Halt the transmission and compare expected and actual data same as we already done done in fifo_en_tx_test 13. When we read all the data the dma_tx_req must HIGH again	DW_apb_uart_ databook section:: 1.2.1 2.13 4.5 6.1 5.1	PASSED			
	☑	dma_rx_test	DMA I/F during receive	To test the DMA operation and DMA RX interface signals i.e. dma_rx_ack_n, dma_rx_req_n, dma_rx_single_n	C-test & sequence	**Configure the IP** (as Step 1-6 in single_tx_test) 7. Write to FCR for enabling FIFO and setting threshold level also called watermark level 7.1. Write 1 to the FCR[0] (FCROE) for enabling FIFO 7.2. Write at FCR[7:6] (RT) bit for RX FIFO trigger level also called watermark level 8. Fill the RX-FIFO by driving "sin" pin of IP more than watermark level 9. IP will request DMA to read from RX-FIFO using DMA I/F signals. Monitor the signal in the respective agent and communicate to c-test via mailbox 10. Read the RBR and communicate via mailbox to the respective agent to drive dma_rx_ack. 11. Repeat step-10 untill the dma_rx_req gets down 12. Compare the actual and expected data as we have done in fifo_en_rx_test **Run the test for different watermark level and observe the underflow and overflow of RX-FIFO also**	DW_apb_uart_ databook section:: 1.2.1 2.13 4.5 6.1 5.1	PASSED			

MODEM Tests	☑	modem_cntrl_txrx_test	TX and RX operations controlled by modem control signals	Disable autoflow control and enable fifo mode, Control the TX operation by enabling DTR and RX operation by enabling RTS too.	C-test	**Configure the IP** (as Step 1-6 in single_tx_test) 7. Write 1 to the FCR[0] (FCROE) for enabling FIFO 8. Disable dtr, rts and autoflow control in MCR 9. Fill the transmit fifo 10. Enable dtr so the vip assert cts and transmission can start 11. Compare expected and actual data same as in fifo_en_tx_test 12. Now enable rts too, so the vip drive the rx-sequence 13. Compare the expected and received payload same as in fifo_en_rx_test	DW_apb_uart_databook section:: 1.2.1 2.10 4.4 5.1 5.1.9 5.1.11	**PASSED**			
Transmission Tests (RS485)	☐	tx_test	Transmission operation by all **randomized** configuration settings	Run multiple transmissions every time with all the randomized configuration settings	C-test	**Configure the IP** (as Step 1-6 in single_tx_test) 7. Enable and program the RS485 mode 8. Randomize the character to be transmitted in THR register 9. Monitor the "sout" pin via VIP according to the configuration settings and compare the expected and the received data 10. Wait for the transmission data to be completely transmitted and scoreboard updated the mailbox 11. Compare the data written to THR and the data received in monitor **Repeat the test to check all RS485 modes and assertion/ deassertion of re & de signal**	DW_apb_uart_databook section:: 1.2.1 6.1 5.1.8 5.1.2 5.1.4 5.1.3 5.1.7		* May need to add more tests to check all RS485 modes		
Receive Tests (RS485)	☐	rx_test	Receive operation by with all the configuration settings **randomized**	Run number of receive tests every time with all the random configuration settings	C-test & sequence	**Configure the IP** (as Step 1-6 in single_tx_test) 7. Enable and program the RS485 mode 8. Drive payload on "sin" pin via VIP driver w.r.t. the configuration settings. IP will start receiving data as soon the start-bit received at "sin" pin 9. Wait untill the VIP completely drives the payload 10. Read the RBR register 11. Compare the expected payload driven at sout and the received data read from RBR **Repeat the test to check all RS485 modes and assertion/ deassertion of re & de signal**	DW_apb_uart_databook section:: 1.2.1 6.1 5.1.8 5.1.2 5.1.4 5.1.3 5.1.7		* May need to add more tests to check all RS485 modes		
Baud rate check	☐	brate_test	Test the baudrate generator and baudout_n pin	Program the DLL and DLH for baud ticks setting, and monitor baudout_n pin. The baudout_n pin must behaves correspond to the values written in DLL and DLH	C-test	**Configure the IP** (as Step 1-6 in single_tx_test) 7. Enable the fifo and run some transmission test 8. Monitor the baudout_n pin in the respective *agent.monitor* 9. Count the number of clock cycles between the two consecutive negitive edges of baudout_n 10. The above count value must be equal to the value programmed in DLL and DLH registers	DW_apb_uart_databook section:: 1.2.1 2.1 4.8				
General CSR Test	☑	rd_reset_val_test	Check the default value of each CSR after applying reset	Verify all reset values of each CSR as mentioned in the **DW_apb_uart_cc_constants.v**	C-test	Read every CSR value and compare with expected results mentioned in the DW_apb_uart_databook section::	DW_apb_uart_databook section:: 5.1	**PASSED**			
	☑	bit_toggle_test	Toggling of R-W bits of every CSR	Toggle each read-write bit of every CSRs and interface ports with alternating 1's and 0's.	C-test	Loop through each CSR's bit and interface port to write in such a way that every bit toggles. Write all zeros and all ones than alternative ones and zeros		**PASSED**			
	☑	illegal_access_test	check the access policies of every field of each csr	Different Fields of a CSR comes with different access policies, To verify non-aliasing among access order and access policies between the fields of every CSR, we can make wrong access to the fields.	C-test	-Write zeros and ones to the read-only bits of every csr, the value souldn't be updated -Write zeros and ones to the write-only bits of every csr, we should read the default value upon reading		**PASSED**			
	☐										
	☐										