AXI DMA - Test Plan

Test ID	Feature	Test Name	Description	Input Stimulus	Expected	Checking Procedure	Status	Comments/Screenshots
1	**Reset**	reset_test	When reset is asserted, the DMA Core is resetted and no read or write operation can happen unless configured.	Apply clock; axi_resetn = 0; Execute Read_status Sequence	DMA core should read default values for S2MMSR and DMASR Registers. The read value should be 'h1 indicating both channels halted.	Checkers in register Sequences can validate the reset state of the DMA Core. Can also be confirm via waveform.	Pass	
Independent Operation								
2	**MM2S Enable**	mm2s_enable_test	Enables the Memory Mapped to Stream Read Operation. The dut will start reading from the provided base address and streams the data on the AXI-Stream Interface.	Configure the DMA's internal registers as following: 1. MM2S_DMACR = 32'h11001 2. MM2S_SA = 32'h0 3. MM2S_LENGTH = 32'h80	The DMA should assert **arvalid** alongwith valid **SRC_ADDR** to make a read request to AXI Slave Memory.	The signal activity can be monitored from waverform.	Pass	
3	**S2MM Enable/Write Test**	s2mm_enable_test/write_test	Enables the Stream to Memory Mapped Write Operation. The dut will take stream as input and start writing on the provided base address via AXI-4 Interface.	Configure the DMA's internal registers as following: 1. S2MM_DMACR = 32'h11001 2. S2MM_DA = 32'h0 3. S2MM_LENGTH = 32'h80 4. Start Axis_write Sequence	The DMA should be able to accept the write transactions through **tvalid & tready** handshake according to AXI-Stream Protocol.	The signal activity can be monitored from waverform.	Pass	
4	**MM2S READ**	read_test	Activates the Stream for Memory Mapped (MM2S) READ operations and initiates a Read Sequence to obtain the output on Stream. The Read Agent will transfer the data to the scoreboard, which has its internal memory pre-loaded with the same data (as BRAM). The scoreboard will verify each byte received from the DMA against the bytes in our internal memory model, and will report an error if there is any mismatch.	Configure the DMA's internal registers as following: 1. MM2S_DMACR = 32'h11001 2. MM2S_SA = cfg.SRC_ADDR 3. MM2S_LENGTH = cfg.DATA_LENGTH 4. Start Axis_read_sequence	The DMA Should read same data from BRAM satrting form the **SRC_ADDR** as the data read from Memory Model in our scoreboard.	The scoreboard validates each byte received from the DMA against our memory model reads and generates an error if any mismatch is detected.	Pass	
5	**Read After Write Test**	raw_test	Enables the Stream to Memory Mapped Write Operation and then writes 128 Bytes to the provided destination address. The write operation is verified by suceeding MM2S Read Operation.	**Step 1:** Configure the DMA's internal registers as following: 1. S2MM_DMACR = 32'h11001 2. S2MM_DA = 32'h0 3. S2MM_LENGTH = 32'h80 **Step 2:** Wait for Axis_write Sequence to Complete. **Step 3:** Configure the DMA's internal registers as following: 1. MM2S_DMACR = 32'h11001 2. MM2S_SA = 32'h0 3. MM2S_LENGTH = 32'h80 **Step 4:** Start the Axis_read_sequence to read 128 Bytes.	The bytes read from the BRAM via the MM2S channel should match the bytes written through the S2MM channel of the AXI DMA.	The results can be verified by comparing the bytes read with the bytes written using the scoreboard. The scoreboard maintains an internal memory that updates with each write operation on the S2MM channel. This memory is then compared with the bytes read through the MM2S channel.	Pass	
4 KB Boundary Protection								
6	**4 KB Boundary Protection**	boundary_test	The AXI DMA should provide 4 KB address boundary protection (when configured in non-Micro DMA).	**Step 1:** Set the SRC_ADDR inside config class to be around 4KB boundary so that once the transfer initiates, it has to cross the boundary. In my case, it's **'hFFC**. **Step 2:** Configure the DMA's internal registers with mm2s_boundary sequence. **Step 3:** Start the Axis_read_sequence to read **DATA_LENGTH** Bytes.	The DMA will automatically map read requests to a 4 KB boundary. It should first create a read request with 1 beat for the bytes below the next address boundary, allowing it to map the subsequent bytes to a 4 KB address boundary (e.g., 0x1000).	This can be verified through the console by reading the next generated address associated with the 16-beat INCR burst operation. It can also be observed from the waveform.	Pass	
Data Re-Alignment								
7	**Data Re-Alignment**	data_realignment_test	AXI DMA provides byte-level data realignment allowing memory reads and writes starting at any byte offset location.	**Step 1:** Set the **SRC_ADDR** inside config class at a random byte that is not aligned on word-size boundary. 'h33, for instance. **Step 2:** Configure the DMA for MM2S Read Operation with a defined DATA_LENGTH. **Step 3:** Start the Axis_read_sequence to read DATA_LENGTH Bytes.	The DMA should begin reading from the specified byte offset and automatically align them as valid bytes on the AXI-Stream.	The check can be carried out by cross-comparing the DMA's read operation with the memory model in our scoreboard. This verification can also be confirmed through the waveform.	Pass	
AXI4 DATA WIDTHS								
8	**32 bits**	axi_32_test	AXI DMA has primary AXI4 data width support of 32, 64, 128, 256, 512, and, 1,024 bits				Dropped	
9	**64 bits**	axi_64_test					Dropped	
10	**128 bits**	axi_128_test					Dropped	
11	**256 bits**	axi_256_test					Dropped	
12	**512 bits**	axi_512_test					Dropped	
13	**1024 bits**	axi_1024_test					Dropped	
AXI-Stream DATA WIDTHS								
14	**8 bits**	axis_8_test	AXI DMA has primary AXI4-Stream data width support of 8, 16, 32, 64, 128, 256, 512, and, 1,024 bits				Dropped	
15	**16 bits**	axis_16_test					Dropped	
16	**32 bits**	axis_32_test					Dropped	
17	**64 bits**	axis_64_test					Dropped	
18	**128 bits**	axis_128_test					Dropped	
19	**256 bits**	axis_256_test					Dropped	
20	**512 bits**	axis_512_test					Dropped	
21	**1024 bits**	axis_1024_test					Dropped	
Interrupt on Completion (IOC)								
22	**Read Introut**	read_introut_test	DMA asserts **mm2s_introut** signal on the completion of read transfer when the **IOC_IrqEn** bit is set/enabled in MM2S_DMACR register. After assertion, write this bit to 1 to clear it (WC Policy)	1. Enable MM2S Read operation. 2. Execute Axis_read_seq 3. Write this value 'h1000 to MM2S_DMASR Register to clear this interrupt signal.	The dma should assert **mm2s_introut** signal at the end of read transfer. Writing a value of 'h1000 to **MM2S_DMASR** should de-assert this signal.	Can be validated through RAL Model where writing a value of 'h1000 to MM2S_DMASR should clear the interrupt signal. The written data value can be checked against the expected value in Register Sequence. Can also be confirmed through waveform.	Pass	
						Can be validated through RAL Model where writing a value of		

AXI DMA - Test Plan

Test ID	Feature	Test Name	Description	Input Stimulus	Expected	Checking Procedure	Status	Comments/Screenshots
23	**Write Introut**	write_introut_test	DMA asserts **s2mm_introut** signal on the completion of write transfer when the **IOC_IrqEn** bit is set/enabled in S2MM_DMACR register. After assertion, write this bit to 1 to clear it (WC Policy)	1. Enable S2MM WRITE operation. 2. Execute Axis_write_seq 3. Write this value 'h1000 to S2MM_DMASR Register to clear the interrupt signal after the write transfer completion.	The dma should assert **s2mm_introut** signal at the end of write transfer. Writing a value of 'h1000 to **S2MM_DMASR** should de-assert this signal.	'h1000 to S2MM_DMASR should clear the interrupt signal. The written data value can be checked against the expected value in Register Sequence. Can also be confirmed through waveform.	Pass	
Register Bits								
24	**Run/Stop Bits**	rs_test	Run/Stop control for controlling running and stopping of the DMA channel. 0 = Stop; 1 = Run **Note:** For Direct Register mode pending commands/transfers are flushed or completed. AXI4-Stream outs are potentially terminated.	1. Enable MM2S Read operation. 2. Execute Axis_read_seq 3. Write this value 'h0 to MM2S_DMACR Register to clear the R/S bit (Stops DMA Core).	The current transfer should complete, however DMA will not consider any further transfers. i.e. writing to MM2S_LENGTH Register will have no effect.	Can be validated through RAL Model reading the MM2S_DMASR. The halted bit should get asserted after setting the MM2S_DMACR.RS = 0. Can also be confirmed via waveform.	Pass	
25	**Soft Reset**	soft_reset_test	Setting either MM2S_DMACR.Reset = 1 or S2MM_DMACR. Reset = 1 resets the entire AXI DMA engine. After completion of a soft reset, all registers and bits are in the Reset State. • 0 = Normal operation. • 1 = Reset in progress.	1. Enable MM2S Read operation. 2. Execute Axis_read_seq 3. Execute Soft Reset Sequence in parallel with Axis_read seq. 4. At the end, Read Control and Status Registers for validation.	The control and Status Registers should read default values and mm2s_prmry_reset_out_n should assert after **soft_reset.**	The values can be validated through the RAL Model by reading MM2S_DMASR and MM2S_DMACR. The Halted Bit should be asserted after resetting the core. This can also be confirmed through the waveform, where the reset signal should assert and de-assert while the soft_reset is in progress.	Pass	
Halted Bit								
26	**Halted**	halted_write_test	When halted, writing to the LENGTH register has no effect on DMA operations. • 0 = DMA Channel Running • 1 = DMA Channel Halted	1. Execute the Read Registers Sequence. 2. Write to the Length Register using mm2s_length_sequence. 3. Read the Control and Status Registers again using the Read_reg Sequence.	The MM2S channel should remain halted with the Status Register showing a 'h1' value when reading registers. It should not initiate any transactions while halted.	This can be validated through RAL Model Register Reads to check the status of the Control and Status Registers. The MM2S and S2MM Control Registers should read their default values, indicating no change in the channel state and confirming no effect on writing the LENGTH Register when the DMA is halted. This can also be confirmed through the waveform.	Pass	
Idle Bit								
27	**Idle**	idle_state_test	Indicates the state of AXI DMA operations. • 0 = Not Idle (Transfer not Complete) • 1 = Idle (Transfer is Complete)	1. Execute the Read Status Sequence. 2. Then Enable MM2S Read Channel and start Axis_read Sequence. 3. Read the Status Registers again to check for the IDLE Bit.	The Status Register **MM2S_DMASR** ('h04) should read a value of 'h0 while running a transfer, indicating a NOT IDLE state. Once the transfer is complete, the Status Register should read 'h1002, indicating the transfer has finished (IDLE).	This can be validated by reading the status from MM2S_DMASR. It can also be confirmed through the waveform.	Pass	
Interrupt on Error								
28	**Slave Error**	slave_error_test	This error occurs if the slave read from the Memory Map interface issues a Slave Error. • 0 = No Slave Error • 1 = Slave Error Detected	1. Force the **rresp** signal with the slave Error (2'b10) in SLvErr Test class. 2. Then configure the MM2S Read Channel with the Interrupt on Error Enabled (**Err_IrqEn => SlvErr_seq**) and start Axis_read Sequence. 3. Execute the Read the Status Registers.	The mm2s_introut should be generated at the end of transfer to indicate that Error had occured during the last transaction.	The error can also be verified by reading the Status Register (MM2S_DMASR). The rdata[14] and rdata[5] should be asserted indicating a Slave Error..	Pass	
29	**Decode Error**	decode_error_test	This error occurs if the address request points to an invalid address. • 0 = No Decode Error • 1 = Decode Error Detected	1. Hardcode the **rresp signal** with the Decode Error (2'b11) in DUT's instantiation. 2. Then configure the MM2S Read Channel with the Interrupt on Error Enabled (Err_IrqEn => DecErr_seq) and start Axis_read Sequence. 3. Execute the Read the Status Registers.	The **mm2s_introut** should be generated at the end of transfer to indicate that Error had occured during the last transaction.	The error can also be verified by reading the Status Register (MM2S_DMASR). The rdata[14] and rdata[6] should be asserted indicating a Slave Error.	Pass	
Slave Not Ready								
30	**Slave Selection**	buffer_overflow_test	The test aims to determine the size of the DMA's internal buffer by keeping the ready signal low on the Stream Side.	1. Set the DATA_LENGTH to be more than 512 bytes, 640 Bytes for instance. 2. Then configure the MM2S Read Operation with a SRC_ADDR = 'h00 and start Axis_read Sequence.	The DMA should cease producing address requests to the Slave memory once its buffer is full. The buffer reaches capacity after receiving 512 bytes from the slave. Since the tready signal is not asserted, the DMA stores the data in its internal buffer, which fills up after 512 bytes.	Waveform.	Pass	