# SYNCHRONOUS FIFO V1.0.0 – VERIFICATION PLAN

| Label | Design Requirement Description | Stimulus Generation | Functional Coverage | Functionality Check |
|---|---|---|---|---|
| Reset Behavior_1 | After reset "Active low – Asynchronous" is asserted, internal pointers and counters must reset to 0.<br>▪ internal pointers: wr_ptr, rd_ptr<br>▪ counters: count | - rst_n (1 bit) = 0 "asserted"<br>- inputs may be initialized or any random data | - cover all values rst_n (0 or 1) | SVA under requirement condition to check the asyn reset Functionality at every +ve edge of the clk and see internal pointer and count in the next cycle |
| Reset Behavior_2 | After reset "Active low – Asynchronous" is asserted, outputs assigned to Default Values.<br>Output / Default<br>data_out (16 bit) / 0<br>wr_ack (1 bit) / 0<br>overflow (1 bit) / 0<br>full (1 bit) / 0<br>empty (1 bit) / 1<br>almostfull (1 bit) / 0<br>almostempty (1 bit) / 0<br>Underflow (1 bit) / 0 | - rst_n (1 bit) = 0 "asserted"<br>- inputs may be initialized or any random data | - cover all values rst_n (0 or 1) | SVA to check the asyn reset Functionality at every +ve edge of the clk and see all the output in the next cycle |
| Write Acknowledge _3 | When a write enables signal is active and the FIFO is not full, wr_ack (1 bit) should be asserted to confirm the write operation and data_in saved in FIFO location defined by internal pointer wr_ptr (1 bit) then increment pointer. | - wr_en (1 bit) = 1 "asserted"<br>- rd_en (1 bit) = 0 "de-asserted"<br>- rst_n (1 bit) = 1 "de-asserted"<br>- data_in (16 bit) may be any random data | - cover all values wr_en (0 or 1)<br>- cover all values data_in ($2^{16} - 1: 0:$)<br>- cross r_en_cp ,w_en_cp ,wr_ack_cp | SVA under requirement conditions to check the Write Acknowledge at every +ve edge of the clk and see wr_ack in the next cycle |
| Read Behaviour_4 | When a read enables signal is active and the FIFO is not empty, data_out (16 bit) assigned to FIFO location defined by internal pointer rd_ptr (3 bit) then increment pointer. | - wr_en (1 bit) = 0 "de-asserted"<br>- rd_en (1 bit) = 1 "asserted"<br>- rst_n (1 bit) = 1 "de-asserted"<br>- data_in (16 bit) may be any random data | - cover all values rd_en (0 or 1)<br>- cover all values data_out ($2^{16} - 1: 0:$) | ------ |
| Write Pointer_5 | When a write enables signal is active and the FIFO is not full, data_in saved in FIFO location defined by internal pointer wr_ptr (3 bit) then increment pointer.<br>- Cannot exceed the FIFO_DEPTH<br>- When pointer reach FIFO_DEPTH-1 return to zero | - wr_en (1 bit) = 1 "asserted"<br>- rd_en (1 bit) = 0 "de-asserted"<br>- rst_n (1 bit) = 1 "de-asserted"<br>- data_in (16 bit) may be any random data | ------ | ------<br>- SVA Pointer threshold assertion for write_ptr at every positive edge of the clock<br>- SVA Pointer wraparound assertion for write_ptr at every positive edge of the clock |
| Read Pointer_6 | - When a read enables signal is active and the FIFO is not empty, data_out (16 bit) assigned to FIFO location defined by internal pointer rd_ptr (3 bit) then increment pointer.<br>- Cannot exceed the FIFO_DEPTH<br>- When pointer reach FIFO_DEPTH-1 return to zero | - wr_en (1 bit) = 0 "de-asserted"<br>- rd_en (1 bit) = 1 "asserted"<br>- rst_n (1 bit) = 1 "de-asserted"<br>- data_in (16 bit) may be any random data | ------ | ------<br>- SVA Pointer threshold assertion for read_ptr at every positive edge of the clock<br>- - SVA Pointer wraparound assertion for write_ptr at every positive edge of the clock |
| Counter Behaviour_7 | - When a write enables signal is active and the FIFO is not full Counter increment by 1.<br>- Counter does not exceed FIFO_DEPTH. | - wr_en (1 bit) = 1 "asserted"<br>- rd_en (1 bit) = 0 "de-asserted"<br>- rst_n (1 bit) = 1 "de-asserted"<br>- data_in (16 bit) may be any random data | ------ | ------<br>- SVA to check the Counter value at every +ve edge of the clk does not exceed FIFO_DEPTH |
| Counter Behaviour_8 | When a read enables signal is active and the FIFO is not empty, Internal counter - count (16 bit) - decrement by one. | - wr_en (1 bit) = 0 "de-asserted"<br>- rd_en (1 bit) = 1 "asserted"<br>- rst_n (1 bit) = 1 "de-asserted"<br>- data_in (16 bit) may be any random data | ------ | ------ |
| Counter Behaviour_9 | When a read and write enables signal is active and the FIFO is Full, From label 3 wr_ack not not asserted mean write operation not valid operation now and the valid operation is read which make sence so Internal counter - count (16 bit) - decrement by one. | - wr_en (1 bit) = 1 "asserted"<br>- rd_en (1 bit) = 1 "asserted"<br>- rst_n (1 bit) = 1 "de-asserted"<br>- data_in (16 bit) may be any random data | ------ | ------ |
| Counter Behaviour_10 | - When a read and write enables signal is active and the FIFO is Empty, read operation not valid operation as it read empty location may be zero or don't care so valid operation is write which make sence so Internal counter - count (16 bit) - increment by one.<br>- Counter does not exceed FIFO_DEPTH. | - wr_en (1 bit) = 1 "asserted"<br>- rd_en (1 bit) = 1 "asserted"<br>- rst_n (1 bit) = 1 "de-asserted"<br>- data_in (16 bit) may be any random data | ------ | ------<br>- SVA to check the Counter value at every +ve edge of the clk does not exceed FIFO_DEPTH |
| Empty Flag _11 | When the internal count is zero, the empty flag should be asserted. | - | - cover all values empty flag (0 or 1)<br>- cross r_en_cp ,w_en_cp ,empty_cp | SVA under requirement condition to check the empty flag at every +ve edge of the clk |
| Full Flag _12 | - When the internal count equals the FIFO_DEPTH, the full flag should be asserted.<br>- Whenever the FIFO is full, wr_ack is always = 0. | - | - cover all values full flag (0 or 1)<br>- cross r_en_cp ,w_en_cp ,full_cp | - SVA under requirement condition to check the empty flag at every +ve edge of the clk<br>- SVA to check wr_ack at every +ve edge of the clk |
| Almost Full _13 | When the count reaches FIFO_DEPTH - 1, almostfull should be asserted. | - | - cover all values almostfull flag(0 or 1)<br>- cross r_en_cp ,w_en_cp ,almostfull_cp | SVA under requirement condition to check the empty flag at every +ve edge of the clk |
| Almost Empty_14 | When the count equals 1, the almostempty signal should be asserted. | - | - cover all values almostempty flag(0 or 1)<br>- cross r_en_cp ,w_en_cp ,almostempty_cp | SVA under requirement condition to check the empty flag at every +ve edge of the clk |
| Overflow_15 | When a write enables signal is active and the FIFO is full, underflow flag asserted. | - wr_en (1 bit) = 0 "de - asserted"<br>- rd_en (1 bit) = 1 "asserted"<br>- rst_n (1 bit) = 1 "de-asserted"<br>- data_in (16 bit) may be any random data | - cover all values overflow flag(0 or 1)<br>- cross r_en_cp ,w_en_cp ,overflow_cp | SVA under requirement conditions and check at every +ve edge of the clk and see overflow in the next cycle. |
| underflow_16 | When a read enables signal is active and the FIFO is empty, underflow flag asserted. | - wr_en (1 bit) = 1 "asserted"<br>- rd_en (1 bit) = 0 "asserted"<br>- rst_n (1 bit) = 1 "de-asserted"<br>- data_in (16 bit) may be any random data | - cover all values underflow flag(0 or 1)<br>- cross r_en_cp ,w_en_cp ,underflow_cp | SVA under requirement conditions and check at every +ve edge of the clk and see underflow in the next cycle. |