1. class transaction;
3. rand bit oper; // Randomized bit for operation control (1 or 0)
4. bit rd, wr; // Read and write control bits
5. bit [7:0] data\_in; // 8-bit data input
6. bit full, empty; // Flags for full and empty status
7. bit [7:0] data\_out; // 8-bit data output
9. constraint oper\_ctrl {
10. oper dist {1 :/ 50 , 0 :/ 50}; // Constraint to randomize 'oper' with 50% probability of 1 and 50% probability of 0
11. }
13. endclass
15. ///////////////////////////////////////////////////
17. class generator;
19. transaction tr; // Transaction object to generate and send
20. mailbox #(transaction) mbx; // Mailbox for communication
21. int count = 0; // Number of transactions to generate
22. int i = 0; // Iteration counter
24. event next; // Event to signal when to send the next transaction
25. event done; // Event to convey completion of requested number of transactions
27. function new(mailbox #(transaction) mbx);
28. this.mbx = mbx;
29. tr = new();
30. endfunction;
32. task run();
33. repeat (count) begin
34. assert (tr.randomize) else $error("Randomization failed");
35. i++;
36. mbx.put(tr);
37. $display("[GEN] : Oper : %0d iteration : %0d", tr.oper, i);
38. @(next);
39. end -> done;
40. endtask
42. endclass
43. ////////////////////////////////////////////
45. class driver;
47. virtual fifo\_if fif; // Virtual interface to the FIFO
48. mailbox #(transaction) mbx; // Mailbox for communication
49. transaction datac; // Transaction object for communication
51. function new(mailbox #(transaction) mbx);
52. this.mbx = mbx;
53. endfunction;
55. // Reset the DUT
56. task reset();
57. fif.rst <= 1'b1;
58. fif.rd <= 1'b0;
59. fif.wr <= 1'b0;
60. fif.data\_in <= 0;
61. repeat (5) @(posedge fif.clock);
62. fif.rst <= 1'b0;
63. $display("[DRV] : DUT Reset Done");
64. $display("------------------------------------------");
65. endtask
67. // Write data to the FIFO
68. task write();
69. @(posedge fif.clock);
70. fif.rst <= 1'b0;
71. fif.rd <= 1'b0;
72. fif.wr <= 1'b1;
73. fif.data\_in <= $urandom\_range(1, 10);
74. @(posedge fif.clock);
75. fif.wr <= 1'b0;
76. $display("[DRV] : DATA WRITE data : %0d", fif.data\_in);
77. @(posedge fif.clock);
78. endtask
80. // Read data from the FIFO
81. task read();
82. @(posedge fif.clock);
83. fif.rst <= 1'b0;
84. fif.rd <= 1'b1;
85. fif.wr <= 1'b0;
86. @(posedge fif.clock);
87. fif.rd <= 1'b0;
88. $display("[DRV] : DATA READ");
89. @(posedge fif.clock);
90. endtask
92. // Apply random stimulus to the DUT
93. task run();
94. forever begin
95. mbx.get(datac);
96. if (datac.oper == 1'b1)
97. write();
98. else
99. read();
100. end
101. endtask
103. endclass
105. ///////////////////////////////////////////////////////
107. class monitor;
109. virtual fifo\_if fif; // Virtual interface to the FIFO
110. mailbox #(transaction) mbx; // Mailbox for communication
111. transaction tr; // Transaction object for monitoring
113. function new(mailbox #(transaction) mbx);
114. this.mbx = mbx;
115. endfunction;
117. task run();
118. tr = new();
120. forever begin
121. repeat (2) @(posedge fif.clock);
122. tr.wr = fif.wr;
123. tr.rd = fif.rd;
124. tr.data\_in = fif.data\_in;
125. tr.full = fif.full;
126. tr.empty = fif.empty;
127. @(posedge fif.clock);
128. tr.data\_out = fif.data\_out;
130. mbx.put(tr);
131. $display("[MON] : Wr:%0d rd:%0d din:%0d dout:%0d full:%0d empty:%0d", tr.wr, tr.rd, tr.data\_in, tr.data\_out, tr.full, tr.empty);
132. end
134. endtask
136. endclass
138. /////////////////////////////////////////////////////
140. class scoreboard;
142. mailbox #(transaction) mbx; // Mailbox for communication
143. transaction tr; // Transaction object for monitoring
144. event next;
145. bit [7:0] din[$]; // Array to store written data
146. bit [7:0] temp; // Temporary data storage
147. int err = 0; // Error count
149. function new(mailbox #(transaction) mbx);
150. this.mbx = mbx;
151. endfunction;
153. task run();
154. forever begin
155. mbx.get(tr);
156. $display("[SCO] : Wr:%0d rd:%0d din:%0d dout:%0d full:%0d empty:%0d", tr.wr, tr.rd, tr.data\_in, tr.data\_out, tr.full, tr.empty);
158. if (tr.wr == 1'b1) begin
159. if (tr.full == 1'b0) begin
160. din.push\_front(tr.data\_in);
161. $display("[SCO] : DATA STORED IN QUEUE :%0d", tr.data\_in);
162. end
163. else begin
164. $display("[SCO] : FIFO is full");
165. end
166. $display("--------------------------------------");
167. end
169. if (tr.rd == 1'b1) begin
170. if (tr.empty == 1'b0) begin
171. temp = din.pop\_back();
173. if (tr.data\_out == temp)
174. $display("[SCO] : DATA MATCH");
175. else begin
176. $error("[SCO] : DATA MISMATCH");
177. err++;
178. end
179. end
180. else begin
181. $display("[SCO] : FIFO IS EMPTY");
182. end
184. $display("--------------------------------------");
185. end
187. -> next;
188. end
189. endtask
191. endclass
193. ///////////////////////////////////////////////////////
195. class environment;
197. generator gen;
198. driver drv;
199. monitor mon;
200. scoreboard sco;
201. mailbox #(transaction) gdmbx; // Generator + Driver mailbox
202. mailbox #(transaction) msmbx; // Monitor + Scoreboard mailbox
203. event nextgs;
204. virtual fifo\_if fif;
206. function new(virtual fifo\_if fif);
207. gdmbx = new();
208. gen = new(gdmbx);
209. drv = new(gdmbx);
210. msmbx = new();
211. mon = new(msmbx);
212. sco = new(msmbx);
213. this.fif = fif;
214. drv.fif = this.fif;
215. mon.fif = this.fif;
216. gen.next = nextgs;
217. sco.next = nextgs;
218. endfunction
220. task pre\_test();
221. drv.reset();
222. endtask
224. task test();
225. fork
226. gen.run();
227. drv.run();
228. mon.run();
229. sco.run();
230. join\_any
231. endtask
233. task post\_test();
234. wait(gen.done.triggered);
235. $display("---------------------------------------------");
236. $display("Error Count :%0d", sco.err);
237. $display("---------------------------------------------");
238. $finish();
239. endtask
241. task run();
242. pre\_test();
243. test();
244. post\_test();
245. endtask
247. endclass
249. ///////////////////////////////////////////////////////
251. module tb;
253. fifo\_if fif();
254. FIFO dut (fif.clock, fif.rst, fif.wr, fif.rd, fif.data\_in, fif.data\_out, fif.empty, fif.full);
256. initial begin
257. fif.clock <= 0;
258. end
260. always #10 fif.clock <= ~fif.clock;
262. environment env;
264. initial begin
265. env = new(fif);
266. env.gen.count = 10;
267. env.run();
268. end
270. initial begin
271. $dumpfile("dump.vcd");
272. $dumpvars;
273. end
275. endmodule