```
module synchronous_fifo  #(parameter
DEPTH=8, DATA_WIDTH=8)  (
  input clk, rst_n,
  input w_en, r_en,
  input [DATA_WIDTH-1:0] data_in,
  output reg [DATA_WIDTH-1:0] data_out,
  output full, empty
);

  reg [$clog2(DEPTH)-1:0] w_ptr, r_ptr;
  reg [DATA_WIDTH-1:0] fifo[DEPTH];

  // Set Default values on reset.
  always@(posedge clk) begin
    if(!rst_n) begin
      w_ptr <= 0; r_ptr <= 0;
      data_out <= 0;
    end
  end
```

```
// To write data to FIFO
  always@(posedge clk) begin
    if(w_en & !full)begin
      fifo[w_ptr] <= data_in;
      w_ptr <= w_ptr + 1;
    end
  end

  // To read data from FIFO
  always@(posedge clk) begin
    if(r_en & !empty) begin
      data_out <= fifo[r_ptr];
      r_ptr <= r_ptr + 1;
    end
  end

  assign full = ((w_ptr+1'b1) == r_ptr);
  assign empty = (w_ptr == r_ptr);
endmodule
```

```
module sync_fifo_TB;
parameter DATA_WIDTH = 8;

reg clk, rst_n;
reg w_en, r_en;
reg [DATA_WIDTH-1:0] data_in;
wire [DATA_WIDTH-1:0] data_out;
wire full, empty;

// Queue to push data_in
reg [DATA_WIDTH-1:0] wdata_q[$], wdata;

synchronous_fifo s_fifo(clk, rst_n, w_en, r_en, data_in, data_out, full, empty);

always #5ns clk = ~clk;

initial begin
  clk = 1'b0; rst_n = 1'b0;
  w_en = 1'b0;
  data_in = 0;

  repeat(10) @(posedge clk);
  rst_n = 1'b1;

  repeat(2) begin
    for (int i=0; i<30; i++) begin
      @(posedge clk);
      w_en = (i%2 == 0)? 1'b1 : 1'b0;
      if (w_en & !full) begin
        data_in = $urandom;
        wdata_q.push_back(data_in);
      end
    end
    #50;
```

```
    end
  end

  initial begin
    clk = 1'b0; rst_n = 1'b0;
    r_en = 1'b0;

    repeat(20) @(posedge clk);
    rst_n = 1'b1;

    repeat(2) begin
      for (int i=0; i<30; i++) begin
        @(posedge clk);
        r_en = (i%2 == 0)? 1'b1 : 1'b0;
        if (r_en & !empty) begin
          #1;
          wdata = wdata_q.pop_front();
          if(data_out !== wdata) $error("Time = %0t: Comparison Failed: expected
wr_data = %h, rd_data = %h", $time, wdata, data_out);
          else $display("Time = %0t: Comparison Passed: wr_data = %h and rd_data
= %h",$time, wdata, data_out);
        end
      end
      #50;
    end

    $finish;
  end

  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
  end
endmodule
```