**FIFO Memory Implementation in Verilog**

**Code :**

**module fifo (**

**input wire clk, // Clock signal**

**input wire rst\_n, // Active-low reset**

**input wire write\_en, // Write enable**

**input wire read\_en, // Read enable**

**input wire [7:0] data\_in, // Input data (8-bit UART data)**

**output reg [7:0] data\_out, // Output data**

**output wire fifo\_full, // FIFO full flag**

**output wire fifo\_empty, // FIFO empty flag**

**output wire [4:0] fifo\_count // Number of stored elements**

**);**

**// Parameters**

**parameter FIFO\_DEPTH = 6;**

**parameter DATA\_WIDTH = 8;**

**// Internal memory (FIFO storage)**

**reg [DATA\_WIDTH-1:0] fifo\_mem [FIFO\_DEPTH-1:0];**

**// Pointers and counter**

**reg [2:0] write\_ptr; // 3 bits enough for depth 6**

**reg [2:0] read\_ptr;**

**reg [4:0] fifo\_count\_reg;**

**// Write operation**

**always @(posedge clk or negedge rst\_n) begin**

**if (~rst\_n) begin**

**write\_ptr <= 3'b0;**

**fifo\_count\_reg <= 5'b0;**

**end else if (write\_en && !fifo\_full) begin**

**fifo\_mem[write\_ptr] <= data\_in;**

**write\_ptr <= (write\_ptr == FIFO\_DEPTH - 1) ? 0 : write\_ptr + 1'b1;**

**fifo\_count\_reg <= fifo\_count\_reg + 1'b1;**

**end**

**end**

**// Read operation**

**always @(posedge clk or negedge rst\_n) begin**

**if (~rst\_n) begin**

**read\_ptr <= 3'b0;**

**data\_out <= 8'b0;**

**end else if (read\_en && !fifo\_empty) begin**

**data\_out <= fifo\_mem[read\_ptr];**

**read\_ptr <= (read\_ptr == FIFO\_DEPTH - 1) ? 0 : read\_ptr + 1'b1;**

**fifo\_count\_reg <= fifo\_count\_reg - 1'b1;**

**end**

**end**

**// FIFO status signals**

**assign fifo\_full = (fifo\_count\_reg == FIFO\_DEPTH);**

**assign fifo\_empty = (fifo\_count\_reg == 0);**

**assign fifo\_count = fifo\_count\_reg;**

**endmodule**

**Test Bench :**

**`timescale 1ns/1ps**

**module fifo\_tb;**

**reg clk;**

**reg rst\_n;**

**reg write\_en;**

**reg read\_en;**

**reg [7:0] data\_in;**

**wire [7:0] data\_out;**

**wire fifo\_full;**

**wire fifo\_empty;**

**wire [4:0] fifo\_count;**

**// Instantiate FIFO**

**fifo uut (**

**.clk(clk),**

**.rst\_n(rst\_n),**

**.write\_en(write\_en),**

**.read\_en(read\_en),**

**.data\_in(data\_in),**

**.data\_out(data\_out),**

**.fifo\_full(fifo\_full),**

**.fifo\_empty(fifo\_empty),**

**.fifo\_count(fifo\_count)**

**);**

**// Clock generation: 10ns period**

**always #5 clk = ~clk;**

**initial begin**

**clk = 0;**

**rst\_n = 0;**

**write\_en = 0;**

**read\_en = 0;**

**data\_in = 8'd0;**

**#10;**

**rst\_n = 1;**

**// Write values: 1, 2, 3, 4**

**@(posedge clk); write\_en = 1; data\_in = 8'd1;**

**@(posedge clk); data\_in = 8'd2;**

**@(posedge clk); data\_in = 8'd3;**

**@(posedge clk); data\_in = 8'd4;**

**@(posedge clk); write\_en = 0;**

**#20;**

**// Read the values**

**repeat(4) begin**

**@(posedge clk); read\_en = 1;**

**@(posedge clk); read\_en = 0;**

**end**

**#20;**

**$finish;**

**end**

**// Monitor**

**initial begin**

**$monitor("Time=%0t | wr\_en=%b rd\_en=%b data\_in=%d data\_out=%d full=%b empty=%b count=%d",**

**$time, write\_en, read\_en, data\_in, data\_out, fifo\_full, fifo\_empty, fifo\_count);**

**end**

**endmodule**