**FIFO**

**VERILOG CODE:**

module fifo\_with\_gray\_operations (

input wire clk,

input wire reset,

input wire wr\_en,

input wire rd\_en,

input wire [3:0] binary\_in,

output reg [3:0] fifo\_data\_out,

output wire and\_result,

output wire or\_result

);

reg [3:0] gray\_in;

reg [3:0] fifo\_mem [1:0]; // Simple 2-depth FIFO for demonstration

reg fifo\_full, fifo\_empty;

integer write\_ptr, read\_ptr;

// Binary to Gray code conversion

always @(\*) begin

gray\_in = binary\_in ^ (binary\_in >> 1);

end

// FIFO Write Operation

always @(posedge clk or posedge reset) begin

if (reset) begin

write\_ptr <= 0;

fifo\_full <= 0;

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

fifo\_mem[write\_ptr] <= gray\_in;

write\_ptr <= write\_ptr + 1;

if (write\_ptr == 1) fifo\_full <= 1;

end

end

// FIFO Read Operation

always @(posedge clk or posedge reset) begin

if (reset) begin

read\_ptr <= 0;

fifo\_empty <= 1;

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

fifo\_data\_out <= fifo\_mem[read\_ptr];

read\_ptr <= read\_ptr + 1;

if (read\_ptr == 1) fifo\_empty <= 1;

end

end

// Simple FIFO control logic to reset pointers

always @(posedge clk or posedge reset) begin

if (reset) begin

fifo\_full <= 0;

fifo\_empty <= 1;

write\_ptr <= 0;

read\_ptr <= 0;

end else if (wr\_en && write\_ptr == 1) begin

fifo\_full <= 1;

fifo\_empty <= 0;

end else if (rd\_en && read\_ptr == 1) begin

fifo\_empty <= 1;

fifo\_full <= 0;

end

end

// Perform bitwise AND and OR operations on the FIFO data

assign and\_result = fifo\_data\_out[3] & fifo\_data\_out[2] & fifo\_data\_out[1] & fifo\_data\_out[0];

assign or\_result = fifo\_data\_out[3] | fifo\_data\_out[2] | fifo\_data\_out[1] | fifo\_data\_out[0];

endmodule