Source Code

```
module fifo( input [7:0] data_input, input clk, rst, op_read, op_write,
                output empty, full, output reg [3:0] counter, output reg [7:0] data_output);
reg [7:0] ram[0:7];
reg [2:0] read_pointer=0, write_pointer=0;
assign empty=(counter==0)?1:0;
assign full=(counter==8)?1:0;


//////////WRITE OPERATION//////////

always @ (posedge clk) begin: write
        if(op_write && !full)
                ram[write_pointer]<= data_input;
        else if(op_write && op_read)
                ram[write_pointer]<= data_input;
end


//////////READ OPERATION//////////

always @ (posedge clk) begin: read
        if(op_read && !empty)
                data_output<= ram[read_pointer];
        else if(op_write && op_read)
                data_output<= ram[read_pointer];
end


//////////POINTER//////////

always @ (posedge clk) begin: pointer
        if(rst) begin
                write_pointer <=0;
                read_pointer <=0;
        end else begin
                write_pointer <= ((op_write && !full) || (op_write && op_read)) ? write_pointer+1 : write_pointer;
                read_pointer <= ((op_read && !empty) || (op_write && op_read)) ? read_pointer+1 : read_pointer;
        end
        if(write_pointer==8)
                write_pointer=0;
        else if(read_pointer==8)
                read_pointer=0;
end
```

```
//////////COUNTER//////////

always @ (posedge clk) begin: count
        if(rst) counter <=0;
        else begin
                case({op_write,op_read})
                        2'b00: counter <= counter;
                        2'b01: counter <= (counter==0) ? 0 : counter-1;
                        2'b10: counter <= (counter==8) ? 8 : counter+1;
                        2'b11: counter <= counter;
                        default: counter <= counter;
                endcase
        end
end

endmodule
```