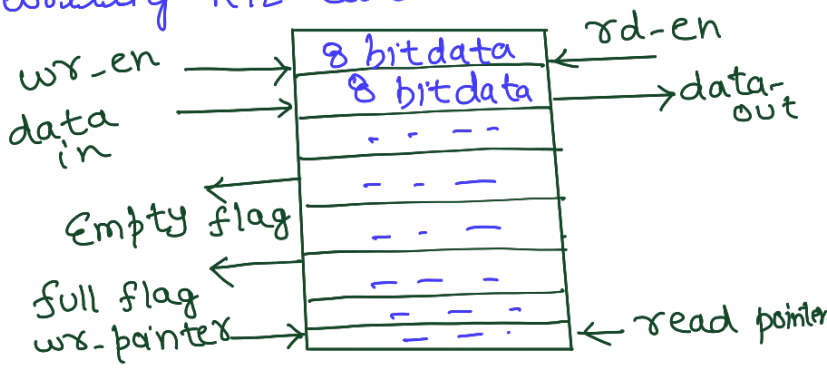
Explanation of RTL Code - (Synchronous FIFO)

```
module fifo(
   input rst, clk, wr_en, rd_en,
   input [7:0] data_in,
   output reg [7:0] data_out,
   output reg empty, full,
   output reg [7:0] counter
);
reg [3:0] rd_ptr, wr_ptr;
reg [7:0] buf_mem[7:0];      //8 bit in 8 location ,8 byte FIFO
```

Internal reg

module name fifo define all port names. First visualize the FIFO buffer structure before writing RTL code.



→ Every location 8-bit

1

```
//Empty and Full logic ,tells wheather fifo is full or empty
always @(counter)
begin
empty <= (counter == 0);
full <= (counter == 8);
end
```

(Full and empty flag logic)

counter counts according to logic and tells if FIFO is empty at full.

Buffer empty → no read can happen
Buffer full → no write can occur.

2 Counter logic. –

```
//Counter logic
always @(posedge clk or posedge rst) begin
if (rst)
        counter <= 0;
else if ((wr_en && !full) && (rd_en && !empty))
        counter <= counter ;
```

To set or manage fifo counter.

```
else if (wr_en && !full)
        counter <= counter + 1;
else if (rd_en && !empty)
        counter <= counter - 1;
else counter <= counter ;
end
```

fifo counter decides when to activate full and empty flag according to this logic

3

```
//Read logic
  always @(posedge clk or posedge rst) begin
    if (rst)
      data_out <= 0;
    else if (rd_en && !empty)
      data_out <= buf_mem[rd_ptr];
  else   data_out <= data_out;
 end
```

To read the data from FIFO.

where rd-ptr is available that data will be retrieved.

4

```
// Write logic
  always @(posedge clk or posedge rst) begin
     if (wr_en && !full)
       buf_mem[wr_ptr] <= data_in;
        else buf_mem[wr_ptr] <= buf_mem[wr_ptr];
  end
```

To write data in FIFO.

→ data-in to be written in the given location where wr-ptr is available.

4

```
// Pointer logic
  always @(posedge clk or posedge rst) begin
```

```
    if (rst) begin
      wr_ptr <= 0;
      rd_ptr <= 0;
    end else begin
      if (wr_en && !full)
        wr_ptr <= wr_ptr + 1;
            else wr_ptr <= wr_ptr;
      if (rd_en && !empty)
        rd_ptr <= rd_ptr + 1;
            else rd_ptr <= rd_ptr;
    end
  end


endmodule
```

Manage both pointer read and write.

write and read pointer.

In Synchronous FIFO write and read happens at Same clock.