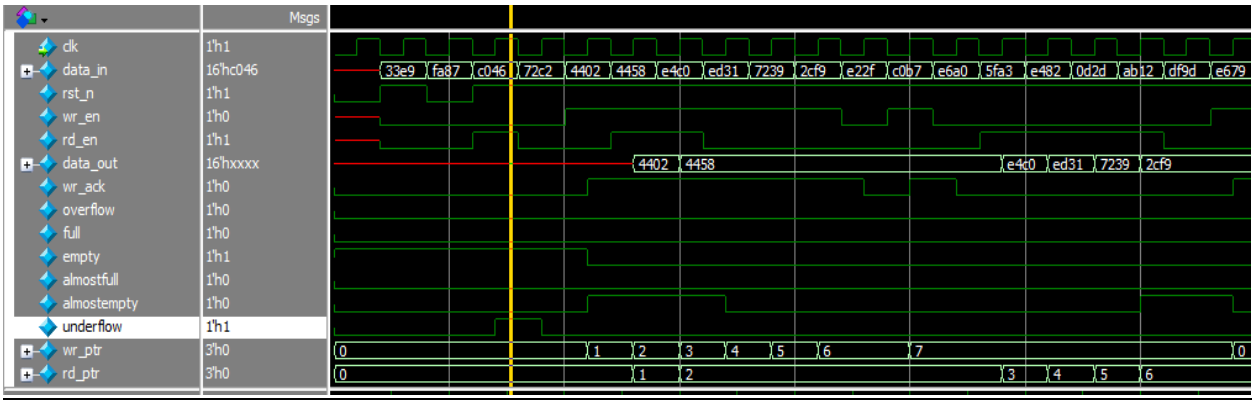
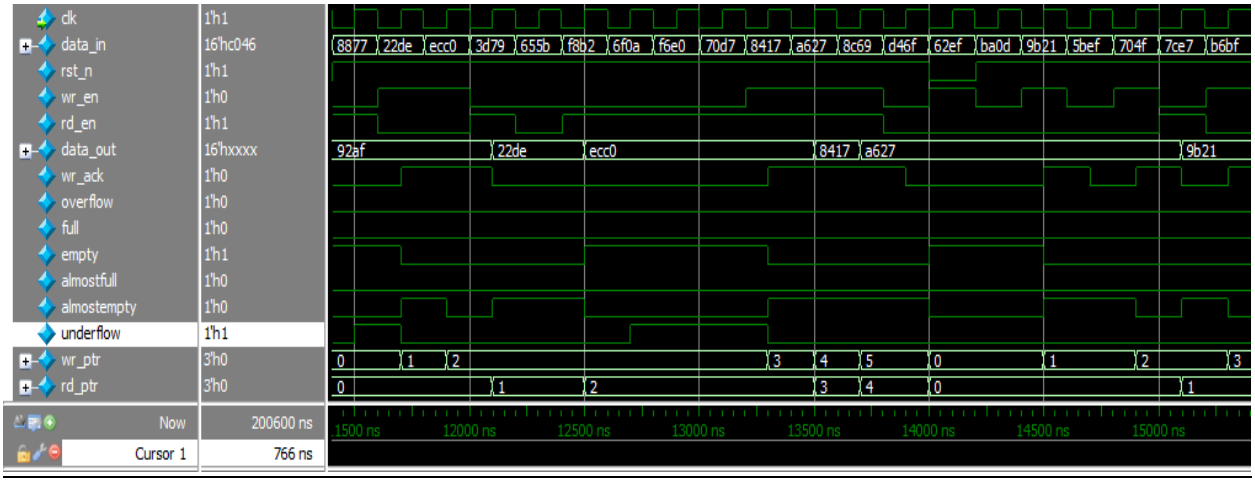
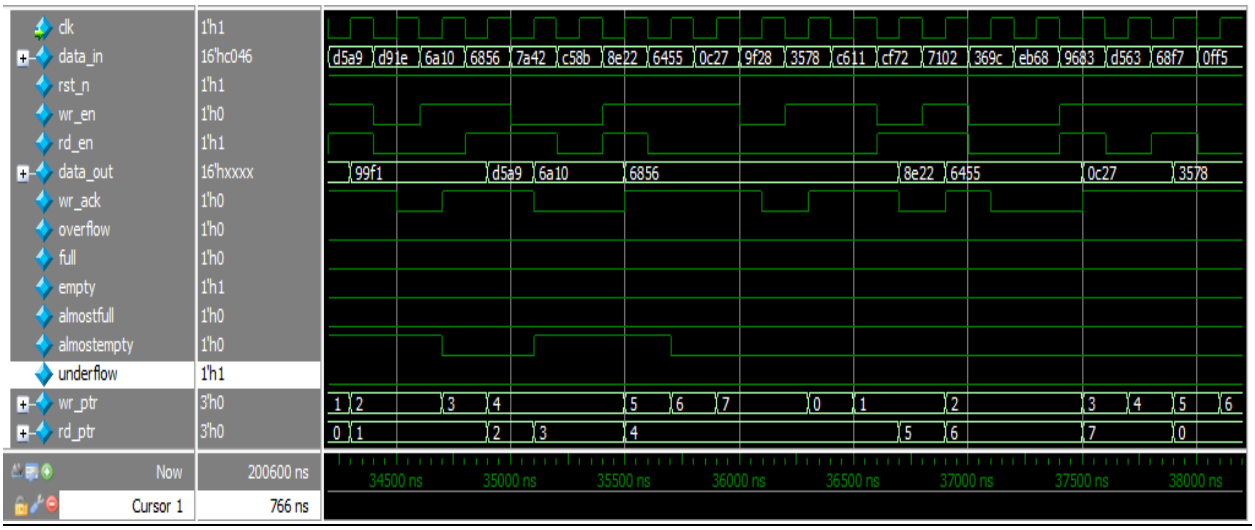
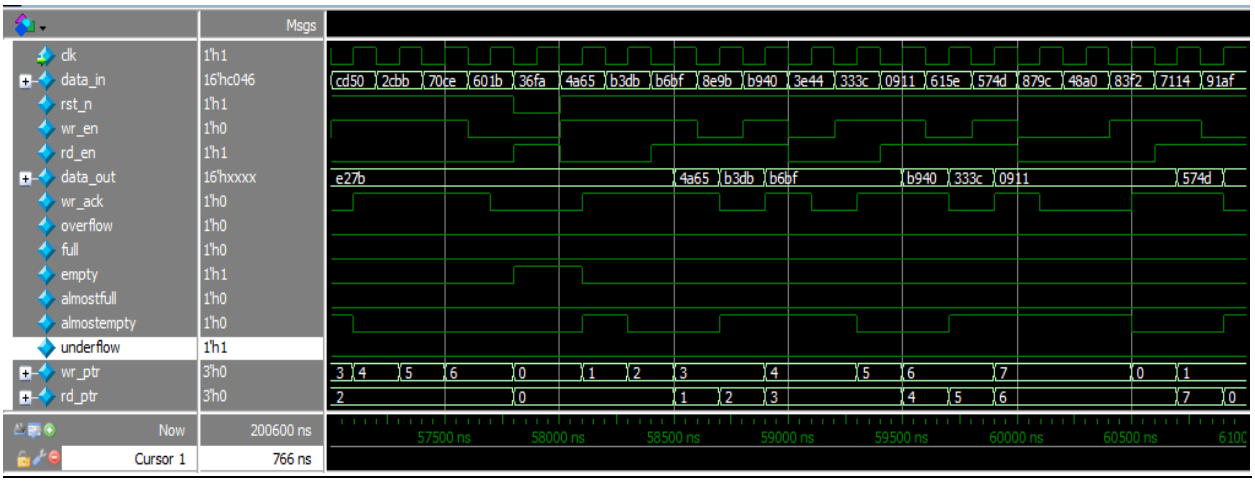
Fifo report

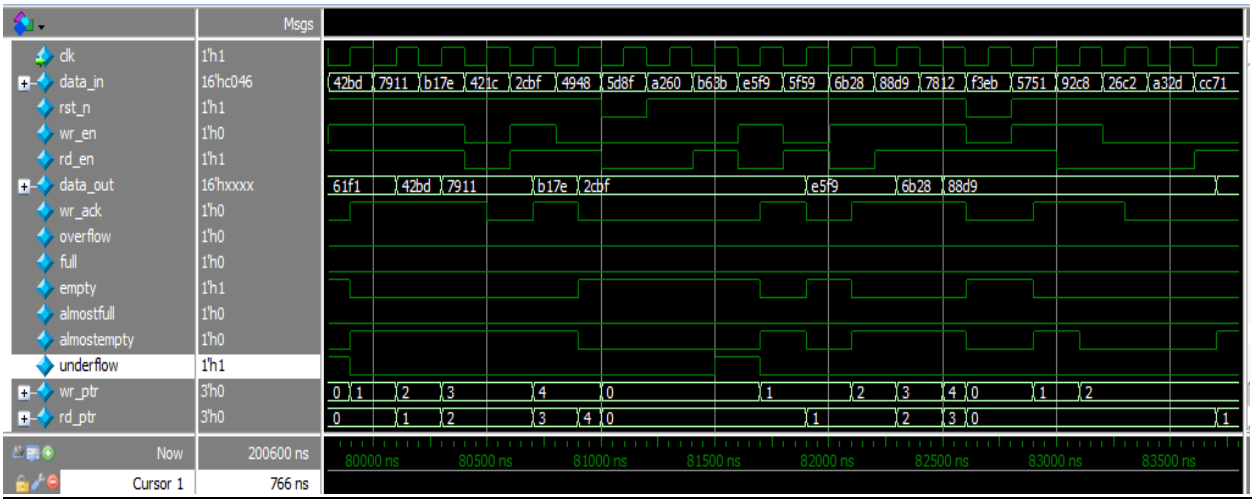
Simulation snippets

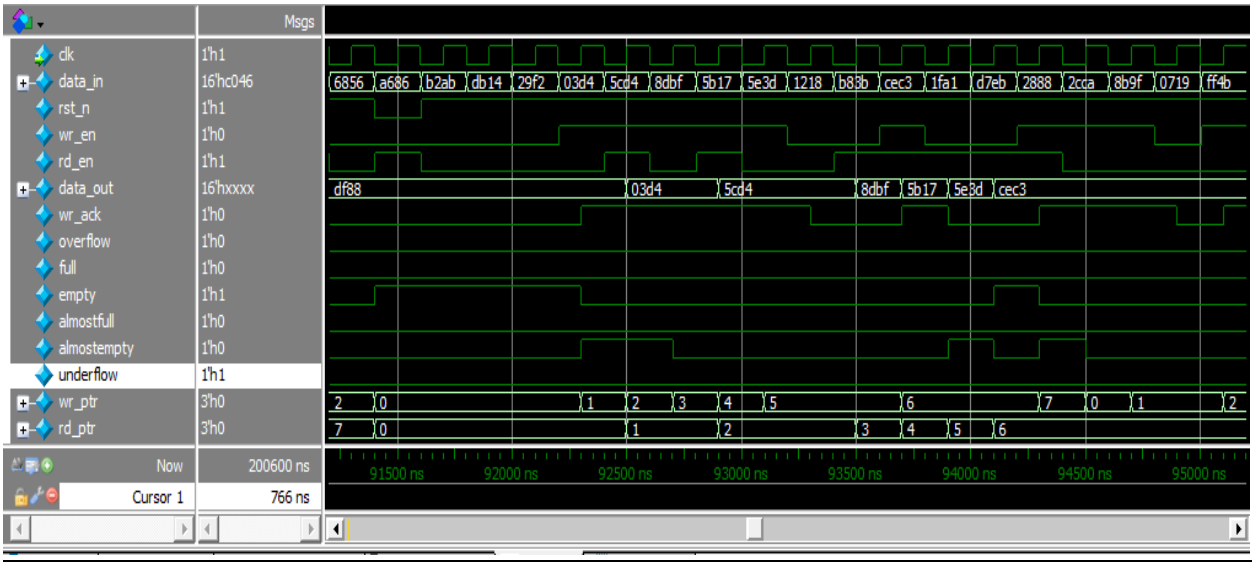







Msgs
clk
data_in
rst_n
wr_en
rd_en
data_out
wr_ack
overflow
full
empty
almostfull
almostempty
underflow
wr_ptr
rd_ptr
Now
Cursor 1
200600 ns
766 ns
cd50 2cbb 70ce 601b 36fa 4a65 b3db b6bf 8e9b b940 3e44 333c 0911 615e 574d 879c 48a0 83f2 7114 91af
e27b 4a65 b3db b6bf b940 333c 0911 574d
57500 ns 58000 ns 58500 ns 59000 ns 59500 ns 60000 ns 60500 ns

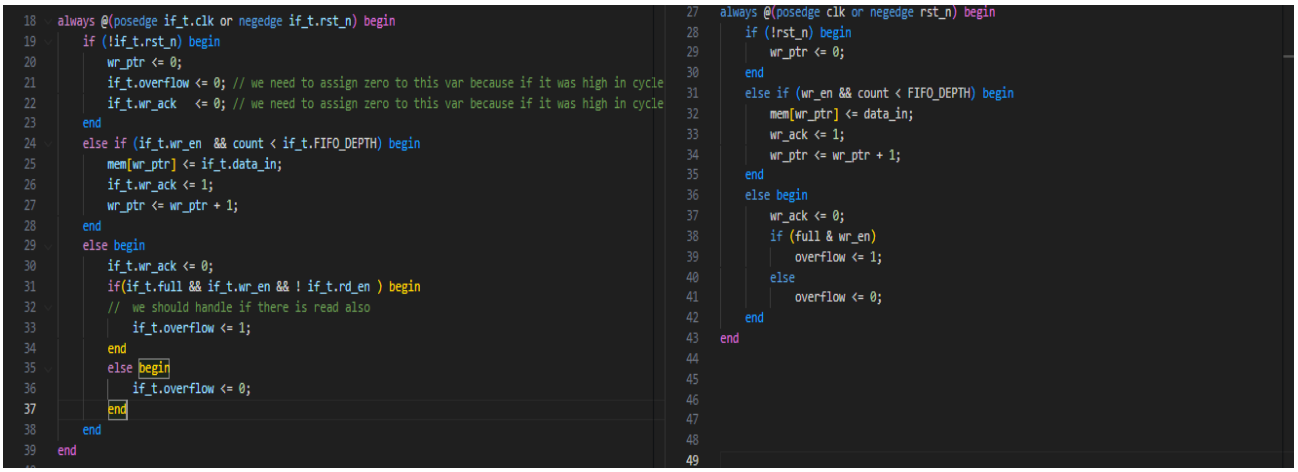

Msgs
clk
data_in
rst_n
wr_en
rd_en
data_out
wr_ack
overflow
full
empty
almostfull
almostempty
underflow
wr_ptr
rd_ptr
Now
Cursor 1
200600 ns
766 ns
42bd 7911 b17e 421c 2cbf 4948 5d8f a260 b63b e5f9 5f59 6b28 88d9 7812 f3eb 5751 92c8 26c2 a32d cc71
61f1 42bd 7911 b17e 2cbf e5f9 6b28 88d9
80000 ns 80500 ns 81000 ns 81500 ns 82000 ns 82500 ns 83000 ns 83500 ns


Msgs
clk
data_in
rst_n
wr_en
rd_en
data_out
wr_ack
overflow
full
empty
almostfull
almostempty
underflow
wr_ptr
rd_ptr
Now
Cursor 1
200600 ns
766 ns
6856 a686 b2ab db14 29f2 03d4 5cd4 8dbf 5b17 5e3d 1218 b83b cec3 1fa1 d7eb 2888 2cca 8b9f 0719 ff4b
df88 03d4 5cd4 8dbf 5b17 5e3d cec3
91500 ns 92000 ns 92500 ns 93000 ns 93500 ns 94000 ns 94500 ns 95000 ns

Detected bugs

1-we need to add overflow and Write Acknowledge to reset because if overflow raise to high and then reset asserted so Overflow still high and same for Acknowledge so we need to put it in Reset

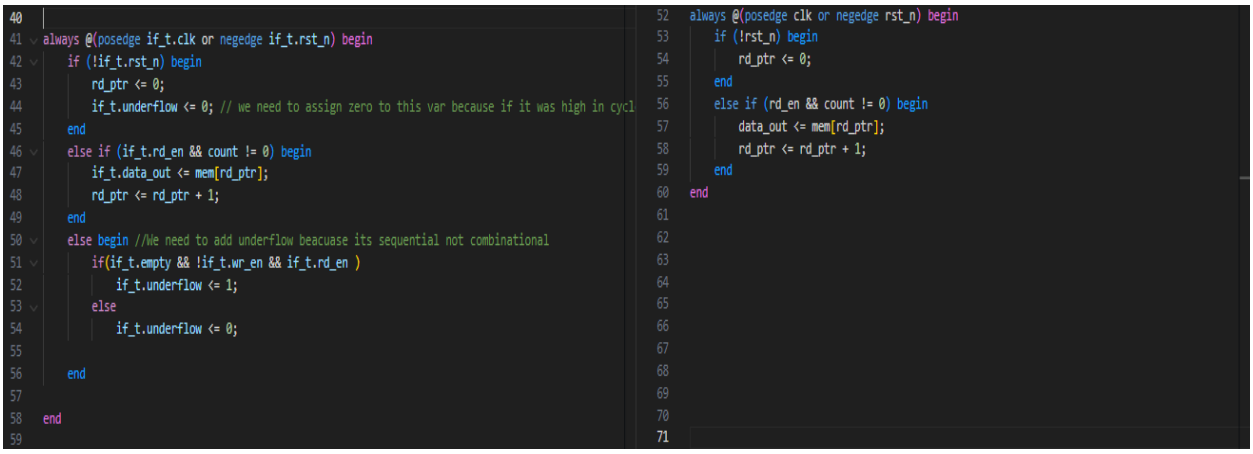
2- we need to put in condition statement in overflow that there is no reading with Writing because if full and write and read are asserted in same time Reading is executed



```
always @(posedge if_t.clk or negedge if_t.rst_n) begin
    if (!if_t.rst_n) begin
        wr_ptr <= 0;
        if_t.overflow <= 0; // we need to assign zero to this var because if it was high in cycle
        if_t.wr_ack   <= 0; // we need to assign zero to this var because if it was high in cycle
    end
    else if (if_t.wr_en  && count < if_t.FIFO_DEPTH) begin
        mem[wr_ptr] <= if_t.data_in;
        if_t.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
    end
    else begin
        if_t.wr_ack <= 0;
        if(if_t.full && if_t.wr_en && ! if_t.rd_en ) begin
        //  we should handle if there is read also
            if_t.overflow <= 1;
        end
        else begin
            if_t.overflow <= 0;
        end
    end
end
```

```
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        wr_ptr <= 0;
    end
    else if (wr_en && count < FIFO_DEPTH) begin
        mem[wr_ptr] <= data_in;
        wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
    end
    else begin
        wr_ack <= 0;
        if (full & wr_en)
            overflow <= 1;
        else
            overflow <= 0;
    end
end
```

3-we need to add underflow flag in Always block because in spec it s sequential



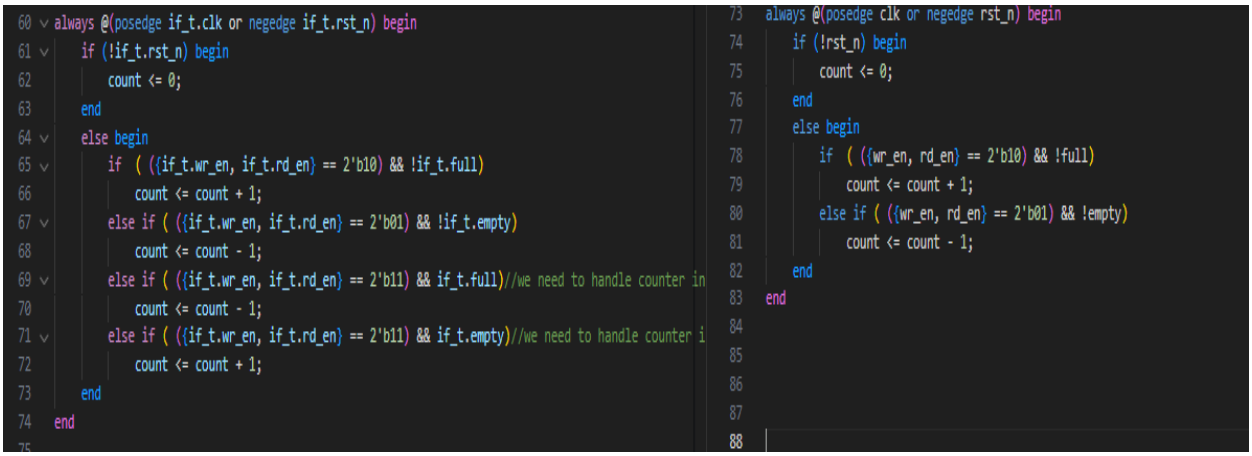
```
always @(posedge if_t.clk or negedge if_t.rst_n) begin
    if (!if_t.rst_n) begin
        rd_ptr <= 0;
        if_t.underflow <= 0; // we need to assign zero to this var because if it was high in cycl
    end
    else if (if_t.rd_en && count != 0) begin
        if_t.data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
    end
    else begin //We need to add underflow beacuase its sequential not combinational
        if(if_t.empty && !if_t.wr_en && if_t.rd_en )
            if_t.underflow <= 1;
        else
            if_t.underflow <= 0;

    end

end
```

```
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        rd_ptr <= 0;
    end
    else if (rd_en && count != 0) begin
        data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
    end
end
```

4-we need to cover additional states that when writing and reading enapled or disabled at same time



```
always @(posedge if_t.clk or negedge if_t.rst_n) begin
    if (!if_t.rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({if_t.wr_en, if_t.rd_en} == 2'b10) && !if_t.full)
            count <= count + 1;
        else if ( ({if_t.wr_en, if_t.rd_en} == 2'b01) && !if_t.empty)
            count <= count - 1;
        else if ( ({if_t.wr_en, if_t.rd_en} == 2'b11) && if_t.full)//we need to handle counter in
            count <= count - 1;
        else if ( ({if_t.wr_en, if_t.rd_en} == 2'b11) && if_t.empty)//we need to handle counter i
            count <= count + 1;
    end
end
```

```
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({wr_en, rd_en} == 2'b10) && !full)
            count <= count + 1;
        else if ( ({wr_en, rd_en} == 2'b01) && !empty)
            count <= count - 1;
    end
end
```