```
module read_logic #(parameter depth = 7, width = 8)(input
clk_out,remove,reset,syn_flush,input [depth: 0]w2rsync_ff2, output reg
[depth-1 : 0]read_addr, output reg [depth : 0]rptr, output reg
read_enable, empty );
parameter[1:0] idle=2'b00, s0=2'b01,s1=2'b11;
reg[1:0] current_state, next_state;
always@(posedge clk_out, negedge reset)
begin
      if(!reset)
            current_state<=idle;
      else if(syn_flush)
            current_state<=idle;
      else
            current_state<=next_state;
end
always@(*)
begin


      case(current_state)
            idle:
                  if(remove)
                        next_state=s0;
                  else
                        next_state=idle;
            s0:
                  if(remove)
                        next_state=s0;
                  else
                        next_state=s1;
            s1:
                  if(remove)
                        next_state=s0;
                  else
                        next_state=s1;

            default:
                        next_state=idle;
      endcase
end


always@(posedge clk_out, negedge reset)
begin
      if(!reset)
      begin
            empty<=1'b1;
            read_enable <= 0;
            read_addr <= 0;
            rptr <=0;
      end
      else if(syn_flush)
      begin
            empty<=1'b1;
```

```
            read_enable <= 0;
            read_addr <= 0;
            rptr <=0;
      end
      else
      case(current_state)
            idle:
            begin
                  if(w2rsync_ff2[depth:0]>0)
                        empty<=1'b0;
                  else
                  begin
                        empty<=1'b1;
                        read_enable <= 0;
                        read_addr <= 0;
                        rptr <= 0;
                  end
            end
            s0:
            begin
                  if(rptr[depth:0] == w2rsync_ff2[depth:0])
                  begin
                        empty<=1'b1;
                        read_enable <= 1'b0;
                        read_addr <= read_addr;
                        rptr <= rptr;
                  end

                  else
                  begin
                        read_addr<=rptr[width-2:0];
                        empty<=1'b0;
                        read_enable<=1'b1;
                        if(rptr==8'b11111111)
                              rptr<=8'h0;
                        else
                              rptr<=rptr+1;
                  end

            end
            s1:
            begin
                  if(rptr[depth:0] !=w2rsync_ff2[depth:0])
                        empty<=1'b0;
            else
            begin
                  empty<=empty;
                  rptr<=rptr;
                  read_enable<=1'b0;
                  read_addr<=read_addr;

            end
            end
endcase
```

```
end
endmodule
```