```
// Created by prof. Mingu Kang @VVIP Lab in UCSD ECE department
// Please do not spread this code without permission
module l0 (clk, in, out, rd, wr, o_full, reset, o_ready);

  parameter row  = 8;
  parameter bw = 4;

  input  clk;
  input  wr;
  input  rd;
  input  reset;
  input  [row*bw-1:0] in;
  output [row*bw-1:0] out;
  output o_full;
  output o_ready;

  wire [row-1:0] empty;
  wire [row-1:0] full;
  reg [row-1:0] rd_en;
//  wire [row-1:0] wr_en;
 //[row-1:0] wr_en;
integer k;
genvar i;

 // assign o_ready = ~o_full  ;
//  assign o_full  = full[0] || full[1] || full[2] || full[3] || full[4] || full[5] || full[6] || full[7] ;
assign o_ready = !full[0] && !full[1] && !full[2] && !full[3] && !full[4] && !full[5] && !full[6] && !full[7]
;
assign o_full = ~o_ready;

  for (i=0; i<row ; i=i+1) begin : row_num
          fifo_depth64 #(.bw(bw)) fifo_instance (
    .rd_clk(clk),
    .wr_clk(clk),
    .rd(rd_en[i]),
    .wr(wr),
          .o_empty(empty[i]),
          .o_full(full[i]),
    .in(in[bw*(i+1)-1:bw*i]),
    .out(out[bw*(i+1)-1:bw*i]),
          .reset(reset));
  end
```

```
  always @ (posedge clk) begin
   if (reset) begin
          rd_en <= 8'b00000000;
   end
   else

          ///////////////// version1: read all row at a time /////////////////
   if(rd == 1)
      begin
//          rd_en <= 8'b11111111;
//end
          ////////////////////////////////////////////////////
          rd_en[0] <= rd;
          rd_en[1] <= rd_en[0];
          rd_en[2] <= rd_en[1];
          rd_en[3] <= rd_en[2];
          rd_en[4] <= rd_en[3];
          rd_en[5] <= rd_en[4];
          rd_en[6] <= rd_en[5];
          rd_en[7] <= rd_en[6];
          end
          else
          begin
          rd_en[0] <= rd;
          rd_en[1] <= rd_en[0];
          rd_en[2] <= rd_en[1];
          rd_en[3] <= rd_en[2];
          rd_en[4] <= rd_en[3];
          rd_en[5] <= rd_en[4];
          rd_en[6] <= rd_en[5];
          rd_en[7] <= rd_en[6];
          end



          ///////////////// version2: read 1 row at a time /////////////////
          ////////////////////////////////////////////////////

          end

endmodule
```

```
// Created by prof. Mingu Kang @VVIP Lab in UCSD ECE department
// Please do not spread this code without permission
module l0 (clk, in, out, rd, wr, o_full, reset, o_ready);

  parameter row  = 8;
  parameter bw = 4;

  input  clk;
  input  wr;
  input  rd;
  input  reset;
  input  [row*bw-1:0] in;
  output [row*bw-1:0] out;
  output o_full;
  output o_ready;

  wire [row-1:0] empty;
  wire [row-1:0] full;
  reg [row-1:0] rd_en;
//  wire [row-1:0] wr_en;
 //[row-1:0] wr_en;
//integer k;
genvar i;

 // assign o_ready = ~o_full  ;
//  assign o_full  = full[0] || full[1] || full[2] || full[3] || full[4] || full[5] || full[6] || full[7] ;
assign o_ready = !full[0] && !full[1] && !full[2] && !full[3] && !full[4] && !full[5] && !full[6] && !full[7]
;
assign o_full = ~o_ready;

  for (i=0; i<row ; i=i+1) begin : row_num
          fifo_depth64 #(.bw(bw)) fifo_instance (
    .rd_clk(clk),
    .wr_clk(clk),
    .rd(rd_en[i]),
    .wr(wr),
          .o_empty(empty[i]),
          .o_full(full[i]),
    .in(in[bw*(i+1)-1:bw*i]),
    .out(out[bw*(i+1)-1:bw*i]),
          .reset(reset));
  end
```

```
  always @ (posedge clk) begin
   if (reset) begin
          rd_en <= 8'b00000000;
   end
   else

          /////////////// version1: read all row at a time /////////////////
   if(rd == 1)
      begin
          rd_en <= 8'b11111111;
end


          end

endmodule
```

```
// Created by prof. Mingu Kang @VVIP Lab in UCSD ECE department
// Please do not spread this code without permission
module ofifo (clk, in, out, rd, wr, o_full, reset, o_ready, o_valid);

  parameter col  = 8;
  parameter bw = 4;

  input  clk;
  input  [col-1:0] wr;
  input  rd;
  input  reset;
  input  [col*bw-1:0] in;
  output [col*bw-1:0] out;
  output o_full;
  output o_ready;
  output o_valid;

  wire [col-1:0] empty;
  wire [col-1:0] full;
  reg  rd_en;

  genvar i;

  assign o_ready = !full[0] && !full[1] && !full[2] && !full[3] && !full[4] && !full[5] && !full[6] &&
!full[7];
  assign o_full  =  ~o_ready;
  assign o_valid =  !empty[0] && !empty[1] && !empty[2] && !empty[3] && !empty[4] && !empty[5]
&& !empty[6] && !empty[7];

  for (i=0; i<col ; i=i+1) begin : col_num
          fifo_depth64 #(.bw(bw)) fifo_instance (
    .rd_clk(clk),
    .wr_clk(clk),
    .rd(rd_en),
    .wr(wr[i]),
          .o_empty(empty[i]),
          .o_full(full[i]),
    .in(in[bw*(i+1)-1:bw*i]),
    .out(out[bw*(i+1)-1:bw*i]),
          .reset(reset));
  end
```

```
  always @ (posedge clk) begin
   if (reset) begin
           rd_en <= 0;

   end
   else
   begin
     rd_en <= rd;

   end

  end

endmodule
```