`timescale 1ns / 1ps

module fifo\_8(data\_out,full, empty, fifo\_th, fifo\_of, fifo\_uf,clk, rst, wr, rd, data\_in);

input wr, rd, clk, rst;

input[3:0] data\_in;

output[3:0] data\_out;

output full, empty, fifo\_th, fifo\_of, fifo\_uf;

wire [4:0] wr\_pt,rd\_pt;

wire fifo\_we,fifo\_rd;

write\_pointer top1(wr\_pt,fifo\_we,wr,full,clk,rst);

read\_pointer top2(rd\_pt,fifo\_rd,rd,empty,clk,rst);

memory\_array top3(data\_out, data\_in, clk,fifo\_we,fifo\_rd, wr\_pt,rd\_pt);

status\_signal top4(full, empty, fifo\_th, fifo\_of, fifo\_uf, wr, rd, fifo\_we, fifo\_rd, wr\_pt,rd\_pt,clk,rst);

endmodule

module memory\_array(data\_out, data\_in, clk,fifo\_we,fifo\_rd, wr\_pt,rd\_pt);

input[3:0] data\_in;

input clk,fifo\_we,fifo\_rd;

input[4:0] wr\_pt,rd\_pt;

output[3:0] data\_out;

reg[3:0] data\_out2[7:0];

reg[3:0] data\_out;

always @(posedge clk)

begin

if(fifo\_we)

begin

data\_out2[wr\_pt[3:0]] <=data\_in ;

end

else if(fifo\_rd)

begin

assign data\_out = data\_out2[rd\_pt[3:0]];

end

end

endmodule

module read\_pointer(rd\_pt,fifo\_rd,rd,empty,clk,rst);

input rd,empty,clk,rst;

output[4:0] rd\_pt;

output fifo\_rd;

reg[4:0] rd\_pt;

assign fifo\_rd = (~empty)& rd;

always @(posedge clk or negedge rst)

begin

if(~rst) rd\_pt <= 0;

else if(fifo\_rd)

rd\_pt <= rd\_pt + 1;

else

rd\_pt <= rd\_pt;

end

endmodule

module status\_signal(full, empty, fifo\_th, fifo\_of, fifo\_uf, wr, rd, fifo\_we, fifo\_rd, wr\_pt,rd\_pt,clk,rst);

input wr, rd, fifo\_we, fifo\_rd,clk,rst;

input[4:0] wr\_pt, rd\_pt;

output full, empty, fifo\_th, fifo\_of, fifo\_uf;

wire fbit\_comp, overflow\_set, underflow\_set;

wire pointer\_equal;

wire[4:0] pointer\_result;

reg full, empty, fifo\_th, fifo\_of, fifo\_uf;

assign fbit\_comp = wr\_pt[4] ^ rd\_pt[4];

assign pointer\_equal = (wr\_pt[3:0] - rd\_pt[3:0]) ? 0:1;

assign pointer\_result = wr\_pt[4:0] - rd\_pt[4:0];

assign overflow\_set = full & wr;

assign underflow\_set = empty&rd;

always @(\*)

begin

full =fbit\_comp & pointer\_equal;

empty = (~fbit\_comp) & pointer\_equal;

fifo\_th = (pointer\_result[4]||pointer\_result[3]) ? 1:0;

end

always @(posedge clk or negedge rst)

begin

if(~rst) fifo\_of <=0;

else if((overflow\_set==1)&&(fifo\_rd==0))

fifo\_of <=1;

else if(fifo\_rd)

fifo\_of <=0;

else

fifo\_of <= fifo\_of;

end

always @(posedge clk or negedge rst)

begin

if(~rst) fifo\_uf <=0;

else if((underflow\_set==1)&&(fifo\_we==0))

fifo\_uf <=1;

else if(fifo\_we)

fifo\_uf <=0;

else

fifo\_uf <= fifo\_uf;

end

endmodule

module write\_pointer(wr\_pt,fifo\_we,wr,full,clk,rst);

input wr,full,clk,rst;

output[4:0] wr\_pt;

output fifo\_we;

reg[4:0] wr\_pt;

assign fifo\_we = (~full)&wr;

always @(posedge clk or negedge rst)

begin

if(~rst) wr\_pt <= 0;

else if(fifo\_we)

wr\_pt <= wr\_pt + 1;

else

wr\_pt <= wr\_pt;

end

endmodule

`timescale 1ns / 1ps

module \_tb\_8\_bit();

reg clk;

reg rst;

reg wr;

reg rd;

reg [3:0] data\_in;

wire [3:0] data\_out;

wire empty;

wire full;

wire fifo\_th;

wire fifo\_of;

wire fifo\_uf;

fifo\_8 dut (data\_out,full, empty, fifo\_th, fifo\_of, fifo\_uf,clk, rst, wr, rd, data\_in);

initial

begin

rst = 1'b0;

wr = 1'b0;

rd = 1'b0;

data\_in = 4'd0;

clk=0;

forever #5 clk= ~clk;

end

initial

begin

rst = 1;

#100;

rst =0;

repeat(20)

begin

#20

wr = 1'b1;

data\_in = data\_in + 8'd1;

#50

wr = 1'b0;

end

repeat(5)

begin

#20

rd = 1'b1;

#40

rd = 1'b0;

end

end

endmodule