Designing 16X7 FIFO Memory via behavioural modelling in Verilog using Xilinx VIVADO 2018.3 :

## Main code :

module FIFO\_mem(

input clk,

input rst,

input [15:0] din,

output reg [15:0] dout,

input read,

input write,

output reg empty,

output reg full

);

parameter DEPTH = 3, MAX\_COUNT = 3'b111;

reg [(DEPTH-1):0] tail;

reg [(DEPTH-1):0] head;

reg [(DEPTH-1):0] count;

reg [15:0] fifo [0:MAX\_COUNT];

reg sr\_read\_write\_empty;

always@(negedge clk)

begin

if(rst)

sr\_read\_write\_empty <= 1'b0;

else if(read == 1'b1 && write == 1'b1 && empty == 1'b1)

sr\_read\_write\_empty <= 1'b1;

else

sr\_read\_write\_empty <= 1'b0;

end

//READ Operation

always@(negedge clk)

begin

if(rst)

begin

dout=16'd0;

head <= 3'b000;

end

else if(read == 1'b1 && empty == 1'b0)

begin

dout <= fifo[head];

head <= head + 1'b1;

end

end

//WRITE Operation

always@(negedge clk)

begin

if(rst)

tail <= 3'b000;

else if(write == 1'b1 && full == 1'b0)

begin

fifo[tail] <= din;

tail <= tail + 1'b1;

end

end

//COUNTER Operation

always@(negedge clk)

begin

if(rst)

count <= 3'b000;

else

begin

case({read,write})

2'b00: count <= count;

2'b01: if(count != MAX\_COUNT)

count <= count + 1;

2'b10: if(count != 3'b000)

count <= count - 1;

2'b11: if(sr\_read\_write\_empty)

count <= count + 1;

else

count <= count;

default: count <= count;

endcase

end

end

//Memory SIGNAL control

always@(count)

begin

empty <= 1'b0;

full <= 1'b0;

if(count == 3'b000)

empty <= 1'b1;

else if(count == MAX\_COUNT)

full <= 1'b1;

end

endmodule

## Testbench Code 1 :

module test\_FIFO(

);

//Inputs

reg clk;

reg [15:0] din;

reg read;

reg write;

reg rst;

// Outputs

wire [15:0] dout;

wire empty;

wire full;

// Instantiate

FIFO\_mem memory(clk, rst, din, dout, read, write, empty, full);

// Initialize Inputs

initial

begin

clk = 1'b0;

din = 32'h0;

read = 1'b0;

write = 1'b0;

rst = 1'b1;

#60;

rst = 1'b1;

#20;

rst = 1'b0;

write = 1'b1;

din = 16'h0;

#20 din = 16'h11;

#20 din = 16'h12;

#20 din = 16'h13;

#20 din = 16'h14;

#20 din = 16'h15;

#20 begin din = 16'h16; read = 1'b1; end

#20 din = 16'h17;

#20 din = 16'h18;

#20 write = 1'b0;

#140 rst = 1'b1;

#40 $finish;

end

always

#10 clk = ~clk;

endmodule

## Testbench Code 2 :

module test\_FIFO02(

);

//Inputs

reg clk;

reg [15:0] din;

reg read;

reg write;

reg rst;

// Outputs

wire [15:0] dout;

wire empty;

wire full;

// Instantiate

FIFO\_mem memory(clk, rst, din, dout, read, write, empty, full);

// Initialize Inputs

initial

begin

clk = 1'b0;

din = 32'h0;

read = 1'b0;

write = 1'b0;

rst = 1'b1;

#40;

rst = 1'b1;

#20;

rst = 1'b0;

write = 1'b1;

din = 16'h0;

#20 din = 16'h11;

#20 din = 16'h12;

#20 din = 16'h13;

#20 din = 16'h14;

#20 din = 16'h15;

#20 din = 16'h16;

#20 din = 16'h17;

#20 din = 16'h18;

#20 begin write = 1'b0;read = 1'b1; end

#180 rst = 1'b1;

#60 $finish;

end

always

#10 clk = ~clk;

endmodule