CALIFORNIA STATE UNIVERSITY, SACRAMENTO

EEE 273

HEIRARCHICAL DIGITAL DESIGN METHODOLOGY

INSTRUCTOR: Dr.BEHNAM ARAD

TERM PROJECT: PARAMETERIZED FIFO BUFFER
(FALL 2016)

TEAM#4: Kiran Bandri Anand (kirananand232@gmail.com)

Greeshma Shailendra (gr.shailen@gmail.com)

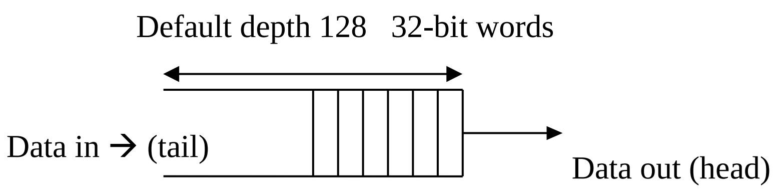
DATE OF SUBMISSION: 12/01/2016

TABLE OF CONTENTS

INTRODUCTION:

The objective of this project is to design, validate and synthesize a parameterized first-in first-out (FIFO) buffer in Verilog. The default depth of the FIFO is 128 32-bit words. It supports flush, insert, and remove operations. During the insert operation, a new word is added to the tail (end) of the buffer and an internal write pointer is incremented on positive edge of clk_in clock. The data at the head of the FIFO is read during the remove operation, and an internal read pointer is incremented on positive edge clk_out clock. A flush clears all words in the buffer. The buffer is considered empty after a flush. It comprises of the following ports:

Ports	Description
1] reset	Every sequential element, content of the FIFO is cleared. It is asynchronous.
2] flush	When asserted, the FIFO is cleared, and the read/write pointers are reset.
3] insert	When asserted, data on data_in port is stored at the tail of the FIFO at clk_in.
4] remove	When asserted, data at the head of the FIFO is removed & placed on the data_out.
5] data_in	Input data port.
6] data_out	Output data port.
7] full	Asserted when the buffer is full and there is no more room for data at clk_in.
8] empty	Asserted when the buffer is empty at clk_out.



CSc/EEE 273
Term Project Status Report
Team #4
Date:12-01-2016

Names	*Signatures*	*Overall Grade*
1.Kiran Bandri Anand		
2.Greeshma Shailendra		

Provide the required information as accurately as possible based on the status of your FIFOsubmitted on the due date. The completed form must appear after Table of Contents (TOC) in your report.

I. Project Report / Demo / 200 points

Table 1: (*To be filled by the instructor*)

Item in the report	Issues	points
Cover sheet		
TOC		
Project Status Report		
Block diagrams for the DUT and Testbench		
FSM diagrams		
Source code		
Test bench • DUT instant • Model • Automated validation • Simulation results (submitted as part of the softcopy only)		
Synthesis 1. Script 2. Synthesis report 3. check design report		
Tabulated area and timing results		

Synthesis reports		
Code Coverage results		
VALIDITY OF RESULTS IN THE REPORT		

II.Design and Modeling Phase: /550 points

Table 2. Source code: Comment on the functionality of the source code you developed for each component of the FIFO as accurately as possible. The comments you provide here must be based on your simulation results. Add more rows as needed. (300 points)

Component	Name of the person who modeled and validated the component	Is this component fully functional?	State any functional issue
Write logic	Kiran	Yes	-
Read logic	Kiran	Yes	-
Memory	Greeshma	Yes	-
Synchronizer	Greeshma	Yes	-
Top module	Kiran	Yes	-
Verification module	Greeshma	Yes	-
Ramdomized testbench	Kiran	Yes	-
Synthesis	Kiran	Yes	-

Table 3 Top-level Design. Comment on the functionality of the FIFO you developed (The functionality of the top-level design). 250 points

State functional issues in DUT	No issues found
State functional issues in the testbench	No issues found

Table 4. Code Coverage: Fill the following table based on the coverage reports vcs generated for your FIFO design for applicable options.

Coverage Type	Percentage	Comments if any
Line	90.70	-
Toggle	98.50	-
Conditional	90.91	-
FSM	93.37	-

III. Synthesis: 25 %

Table 5. Fill out the following table based on your best synthesis trials: timing & area. In each case, state the sign of the timing and area parameters as provided by the Design Compiler tool. The first row is a summary your results based on given clock rates. You should use the following constrains in all synthesis attempts using high synthesis effort:

Max input/output delay based on clk_in	0.2ns
Min input/output delay based on clk_in	0.02ns
Max input/output delay based on clk_out	0.1ns
Min input/output delay based on clk_out	0.01ns

Trial	Clk_in Period in ns	Clk_out Period in ns	Area slack from area report	Timing slack from timing report	data required time for max path from timing report	data arrival time for the max path from timing report path
1.	2 ns	4 ns	-247358.20	Clk_in 0.02 Clk_out 0.83	1.91	1.89
2.	2 ns	4 ns	-315471.94	Clk_in -7.37 Clk_out -3.48	1.91	9.29

Briefly describe any RTL changes you made to improve synthesis results.

Trial 2:

Trial 1 had a memory block which was sequential. To meet a positive slack in synthesis, we designed a combinational memory as part of a second trial and achieved a positive slack.

IV FIFO performance …………/50 points

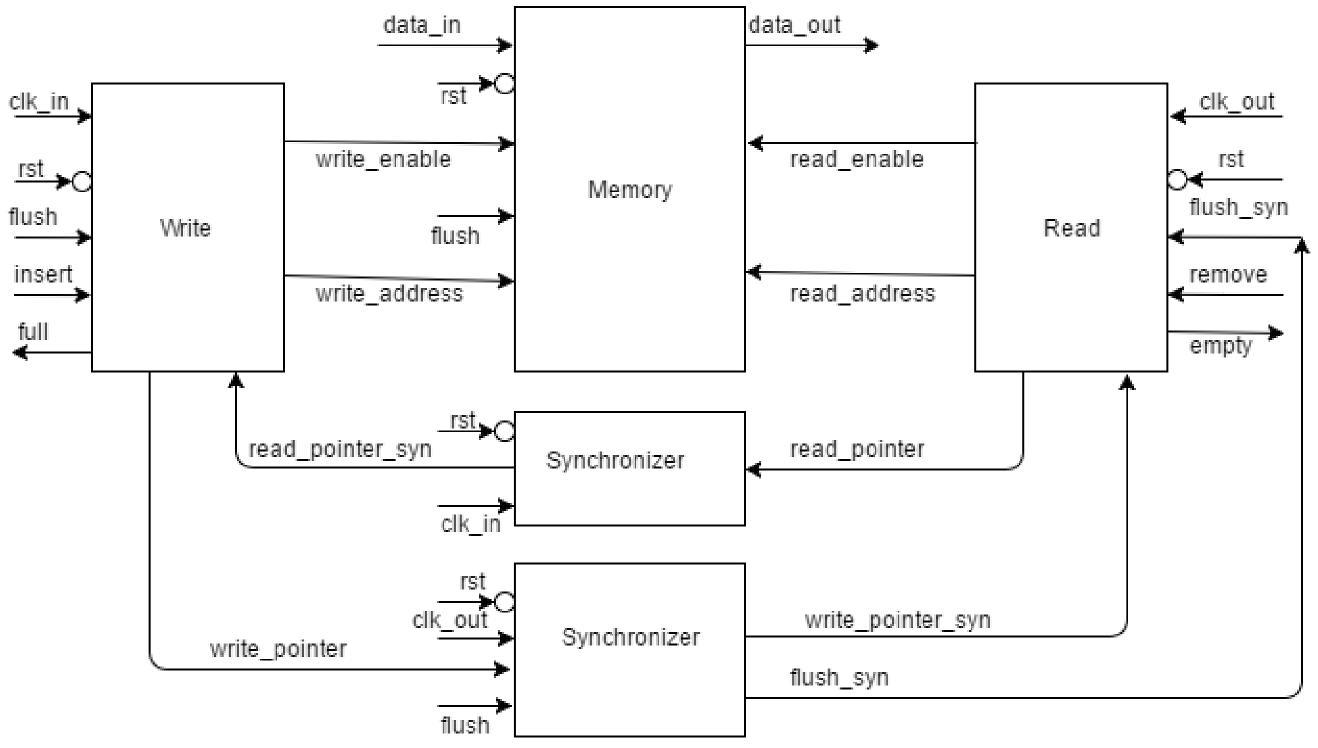
1. What's the maximum clock rate your complete design can operate at?

   clock in performance = 505.05MHz.
   clock out performance=315.45MHz.

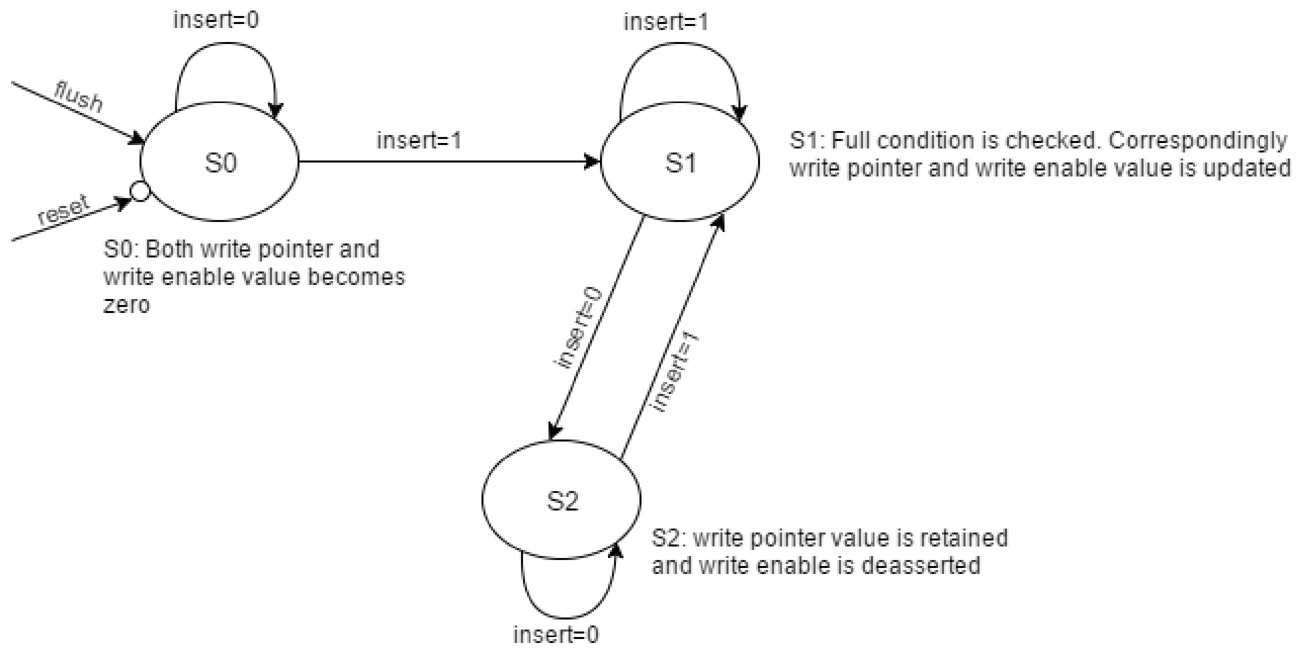
Tables 6: For each partner, state the contribution percentage for each task listed below: Please only provide a percentage.

Name	Design	Simulation	Synthesis	Project report
Kiran Bandri Anand	60%	60%	45%	40%
Greeshma Shailendra	40%	40%	55%	60%

Design Block Diagram:

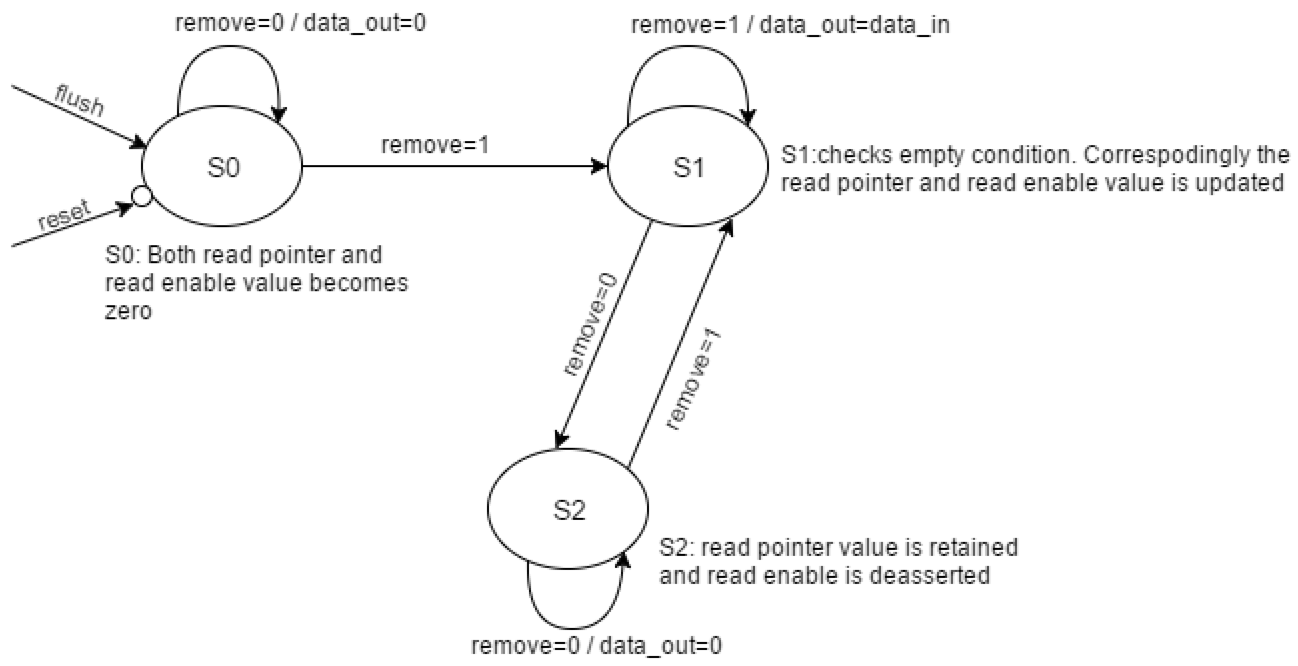


FSM for write logic:

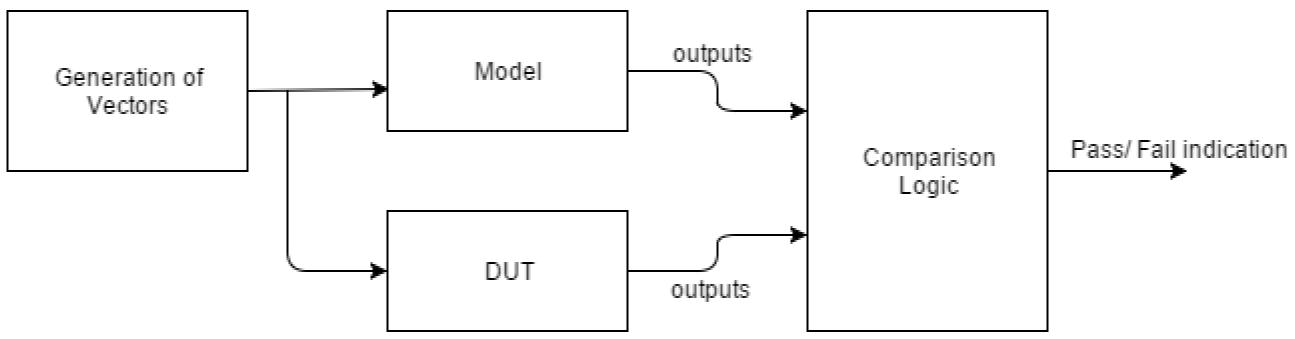


insert=0
insert=1
flush
reset
S0
insert=1
S1
S1: Full condition is checked. Correspondingly write pointer and write enable value is updated
S0: Both write pointer and write enable value becomes zero
insert=0
insert=1
S2
S2: write pointer value is retained and write enable is deasserted
insert=0

FSM for read logic:


remove=0 / data_out=0
remove=1 / data_out=data_in
flush
reset
S0
remove=1
S1
S1:checks empty condition. Correspodingly the read pointer and read enable value is updated
S0: Both read pointer and read enable value becomes zero
remove=0
remove=1
S2
S2: read pointer value is retained and read enable is deasserted
remove=0 / data_out=0

Verification Block Diagram:



SOURCE CODE

a. Memory:

```
//memory module

module memory1 #(parameter M=32,N=128,C=6)(input
rst,flush,write_enable,read_enable,input[M-1:0] temp_data_in,input [C:0]
write_address,read_address,output reg [M-1:0] data_out);
reg [M-1:0] mem[N-1:0];
integer i;
reg [M-1:0]temp;

always @ (*)
begin
   if (!rst)
   begin
        data_out=32'h0;
      for (i=0; i < 128; i=i+1)
      begin
         mem[i] = 32'h0;
      end
   end

   else if(flush)
     begin
      data_out = 32'h0;
      for(i=0; i < N; i=i+1)
      mem[i] = 32'h0;
     end

   else if(write_enable && read_enable)        //write and read operation
   begin
        mem[write_address] = temp_data_in;
      data_out = mem[read_address];
   end

   else if (write_enable)             //write operation
```

```
    begin
        mem[write_address] = temp_data_in;
    end

    else if (read_enable)              //read operation
    begin
     data_out = mem[read_address];
    end

end
endmodule
```

b. Write Logic:

```
//write module

module write#(parameter P=7,C=6)(input
clk_in,rst,flush,insert,input[P:0] read_pointer_syn,output reg
full,write_enable,output reg [C:0] write_address,output reg
[P:0]write_pointer);
parameter S0=2'b00,S1=2'b01,S2=2'b11;
reg[1:0]  CS,NS;

always @(posedge clk_in or negedge rst)
 begin
          if(!rst)
            CS<=S0;
          else if(flush)
          CS<=S0;
          else
            CS<=NS;
 end

always @(*)
 begin
          case(CS)
            S0:if(insert)                    //idle state
                 NS<=S1;
            else
              NS<=S0;
          S1:if(insert)                   //write state
              NS<=S1;
            else
              NS<=S2;
          S2:if(insert)          //state designed to retain previous state outputs
              NS<=S1;
            else
              NS<=S2;
          default:NS<=S0;
      endcase
 end

always @ (posedge clk_in or negedge rst)
 begin
        if(!rst)
        begin
        full<=0;
          write_pointer<=0;
        write_enable<=0;
        write_address<=0;
        end

      else if(flush)
        begin
        full<=0;
          write_pointer<=0;
        write_enable<=0;
        write_address<=0;
        end

        else
        case(CS)

    S0:begin
          full<=0;
```

```
            write_pointer<=0;
            write_enable<=0;
            write_address<=0;
        end

      S1:begin
           if(write_pointer[C:0]==read_pointer_syn[C:0] &&
write_pointer[P]!=read_pointer_syn[P])        //to check full condition
             begin
               full<=1'b1;
               write_pointer<=write_pointer;
             write_enable<=1'b0;
             write_address<=write_address;
             end

             else
             begin
               write_address<=write_pointer[C:0];
               full<=1'b0;
               write_enable<=1'b1;
               if(write_pointer==8'b11111111)
               write_pointer<=8'h0;
               else
                 write_pointer<=write_pointer+1;
             end
         end

      S2:begin
          if(!(write_pointer[C:0]==read_pointer_syn[C:0] &&
write_pointer[P]!=read_pointer_syn[P]))
            begin
              full<=0;
            write_pointer<=write_pointer;
            write_enable<=1'b0;
            write_address<=write_address;
            end
        else
          begin
         full <= full;
          write_pointer<=write_pointer;
          write_enable<=1'b0;
          write_address<=write_address;
          end
          end
   endcase
end
endmodule
```

c. Read Logic:

```
//Read module
module read# (parameter P=7,C=6)(input clk_out, rst, remove, flush_syn,
input[P:0] write_pointer_syn, output reg empty, read_enable, output reg
[C:0] read_address, output reg [P:0] read_pointer);
parameter S0=2'b00,S1=2'b01,S2=2'b11;
reg [1:0] CS,NS;

always @(posedge clk_out or negedge rst)
 begin
          if(!rst)
          CS<=S0;
        else
         if(flush_syn)
           CS<=S0;
          else
             CS<=NS;
 end

always @(*)
 begin
          case(CS)
            S0:if(remove)                //idle state
               NS<=S1;
              else
               NS<=S0;
            S1:if(remove)                //read state
               NS<=S1;
              else
               NS<=S2;
            S2:if(remove)       //state designed to retain previous state outputs
               NS<=S1;
              else
               NS<=S2;
            default:NS<=S0;
        endcase
   end

always @ (posedge clk_out or negedge rst)
 begin
          if(!rst)
          begin
            empty<=1'b1;
          read_pointer<=0;
          read_enable<=0;
          read_address<=0;
          end

        else if(flush_syn)
          begin
          empty<=1'b1;
          read_pointer<=0;
          read_enable<=0;
          read_address<=0;
          end

          else
          case(CS)
          S0:begin
               if(write_pointer_syn[P:0]>0)
                 empty<=1'b0;
               else
               begin
               empty<=1'b1;
               read_pointer<=0;
               read_enable<=0;
               read_address<=0;
               end
             end

       S1:begin
               if(read_pointer[P:0]==write_pointer_syn[P:0])
               begin
                 empty<=1'b1;
                   read_pointer<=read_pointer;
                   read_enable<=1'b0;
                   read_address<=read_address;
               end

               else if (read_pointer[P:0]!=write_pointer_syn[P:0])
               begin
                   read_address<=read_pointer[C:0];
                   empty<=1'b0;
                   read_enable<=1'b1;
                   if(read_pointer==8'b11111111)
                   read_pointer<=8'h0;
                   else
                     read_pointer<=read_pointer+1;
               end
            end

       S2: if(read_pointer[P:0]!=write_pointer_syn[P:0])
              empty<=0;
            else
            begin
              empty<=empty;
               read_pointer<=read_pointer;
               read_enable<=1'b0;
               read_address<=read_address;
```

```
          end
      endcase
 end
endmodule
```

d. Synchronizer Logic:

```
//Synchronizer module

module Syncronizer #(parameter P=7)(input clk_in,clk_out,rst,flush,input
[31:0] data_in,input[P:0]read_pointer,write_pointer,output reg
flush_syn,output reg [P:0]read_pointer_syn,write_pointer_syn,output reg
[31:0] temp_data_in);
reg [P:0]temp1,temp2;
reg temp3;
reg [31:0] temp4;

//Read pointer synchronizer
always @ (posedge clk_in or negedge rst)
 begin
        if(!rst)
        read_pointer_syn<=8'h0;
        else
        begin
        temp1<=read_pointer;
        read_pointer_syn<=temp1;
        end
 end

always @ (posedge clk_in or negedge rst)
 begin
        if(!rst)
        temp_data_in<=32'h0;
        else
        begin
        temp4<=data_in;
        temp_data_in<=temp4;
```

```
         end
 end

//write pointer synchronizer
always @ (posedge clk_out or negedge rst)
 begin
         if(!rst)
         write_pointer_syn<=8'h0;
         else
         begin
         temp2<=write_pointer;
         write_pointer_syn<=temp2;
         end
 end

//flush synchronizer
always @ (posedge clk_out or negedge rst)
 begin
         if(!rst)
         flush_syn <= 1'b0;
         else
         begin
         temp3<=flush;
         flush_syn<=temp3;
         end
 end
endmodule
```

e. Top module:

```
//Top module fifo.v

`include "Syncronizer.v"
`include "memory1.v"
`include "read_logic.v"
`include "write_logic.v"
```

```
module fifo #(parameter N=128,M=32,P=7)(input clk_in, clk_out, flush,
rst, insert,remove,input [M-1:0] data_in,output [M-1:0] data_out,output
full,empty);

wire [P:0]write_pointer_syn, read_pointer_syn, write_pointer,
read_pointer;
wire write_enable, read_enable, flush_syn;
wire [P-1:0]write_address, read_address;
wire [M-1:0] temp_data_in;

//Instantiation of write module
write  #(P,P-1)W1 (.clk_in(clk_in), .rst(rst), .flush(flush), .insert(insert),
.read_pointer_syn(read_pointer_syn), .full(full),
.write_enable(write_enable), .write_address(write_address),
.write_pointer(write_pointer));

//Instantiation of read module
read  #(P,P-1)R1 (.clk_out(clk_out), .rst(rst), .remove(remove),
.flush_syn(flush_syn), .write_pointer_syn(write_pointer_syn),
.empty(empty), .read_enable(read_enable), .read_address(read_address),
.read_pointer(read_pointer));

//Instantiation of memory module
memory1 #(M,N,P-1)M1(.rst(rst), .flush(flush),
.write_enable(write_enable), .read_enable(read_enable),
.temp_data_in(temp_data_in), .write_address(write_address),
.read_address(read_address), .data_out(data_out));

//Instantiation of synchronizer module
Syncronizer #(P)S1(.data_in(data_in), .clk_in(clk_in), .clk_out(clk_out),
.rst(rst), .flush(flush), .flush_syn(flush_syn), .read_pointer(read_pointer),
.write_pointer(write_pointer), .read_pointer_syn(read_pointer_syn),
.write_pointer_syn(write_pointer_syn), .temp_data_in(temp_data_in));

endmodule
```

TEST FIXTURE

a. Verification module:

```
//verification module

module fifo_tb #(parameter f_depth=128,f_width=32,ptr_wdth=7)(input
clk_in,clk_out,flush,reset,insert,remove,input [f_width-1:0]
data_in,output reg [f_width-1:0] data_out,output reg full,empty);
reg [ptr_wdth:0]r_pntr,w_pntr;
reg [ptr_wdth:0]r_pntr,w_pntr;
reg flush_sync,temp1,temp2,temp3;
reg  [f_width-1:0] temp_data_in;
reg [f_width-1:0]f_mem[127:0];
integer i;

always @(posedge clk_in or negedge reset)
begin
  temp_data_in<= data_in;
  if(!reset || flush)
  begin
    empty <= 1;
    data_out<=0;
    full<=0;
    r_pntr <=0;
    w_pntr<=0;
    for(i=0; i<128;i=i+1)
  f_mem[i]<=32'b0;
  end

  else if(insert)
  begin
    if(w_pntr[ptr_wdth-1:0]-1==r_pntr[ptr_wdth-1:0]-1 &&
w_pntr[ptr_wdth]!=r_pntr[ptr_wdth])
    begin
      full<=1'b1;
      w_pntr<=w_pntr;
    end
    else
    begin
      #2 full<=1'b0;
      f_mem[w_pntr[ptr_wdth-1:0]]<= temp_data_in;
      if(w_pntr==8'b11111111)
        w_pntr<=8'h0;
      else
        w_pntr<=w_pntr+1;
    end
  end

  else
  begin
      if(!(w_pntr[ptr_wdth-1:0]-1==r_pntr[ptr_wdth-1:0]-1 &&
w_pntr[ptr_wdth]!=r_pntr[ptr_wdth]))
      full<=0;
    else
    begin
      #2 full<=full;
    w_pntr<=w_pntr;
    end
  end
end

always @(posedge clk_out or negedge reset)
begin
  if(!reset||flush_sync)
  begin
    empty<=1'b1;
  r_pntr<=1'b0;
  data_out<=1'b0;
  end

  else if(remove)
  begin
    if(r_pntr[ptr_wdth:0]-1==w_pntr[ptr_wdth:0]-1)
    begin
```

```
      empty<=1'b1;
      r_pntr<=r_pntr;
      #4 data_out<=data_out;
    end

    else
    begin
      empty<=1'b0;
      #4 data_out<=f_mem[r_pntr[ptr_wdth-1:0]];
      if(r_pntr==8'b11111111)
        r_pntr<=8'h0;
      else
        r_pntr<=r_pntr+1;
    end
  end

  else
  begin
    empty<=empty;
    #4 data_out<=data_out;
    r_pntr<=r_pntr;
  end
end
endmodule
```

<u>b.</u> <u>Automated Validation:</u>

```
//automated validation

`include "testbench_module.v"
`include "fifo.v"
module fifo_fixture;
parameter M=32,N=128;
reg clk_in,clk_out,flush,rst,insert,remove;
reg [M-1:0] data_in;
wire [M-1:0] data_out;
wire full,empty;
wire [M-1:0] data_out1;
wire full1,empty1;
integer i;

//Instantiation of verification module
fifo_tb  #(128,32,7)
u1(clk_in,clk_out,flush,rst,insert,remove,data_in,data_out1,full1,empty1);

//Instantiation of DUT
fifo #(128,32)
u2(clk_in,clk_out,flush,rst,insert,remove,data_in,data_out,full,empty);

initial
$vcdpluson;

initial
begin
  clk_in=1'b0;
  forever #1 clk_in=~clk_in;
end

initial
begin
  clk_out=1'b0;
  forever #2 clk_out=~clk_out;
end

task reset();
  begin
  insert=1'b0;
  remove=1'b0;
  rst=1'b0;
  flush =1'b0;
  #3 rst=1'b1;
  end
endtask

initial begin
```

```
  reset();
  for (i=0; i<150; i=i+1)              //to check full condition
  begin
    @(negedge clk_in);
    data_in=$random;
    insert=1;
    #2;
  end

  insert=0;              //to check empty condition
  remove=1;
  #600;

  insert=0;              //to check flush
  remove=0;
  flush=1;
  #5 flush=0;
  #10;
  for (i=0; i<100; i=i+1)              //random generation of test inputs
  begin
    @(negedge clk_in);
    data_in=$random;
    insert =$random;
    #2;
    insert =0;
    #2;

    @(negedge clk_out);
    remove =$random;
    #4;
    remove=0;
  end

  rst=0;
  #5 rst=1;
  $finish;
end
```

```
initial                                  //checker block
begin
  forever @(posedge clk_out)
  begin
    if(data_out == data_out1)
      $display($time," clk_in=%b, clk_out=%b, reset=%b, flush=%b,
insert=%b, remove=%b, datain=%h,dataout=%h,empty=%b, full=%b
PASS",clk_in,clk_out,rst,flush,insert,remove,data_in,data_out,empty,full);

   else
      $display($time," clk_in=%b, clk_out=%b, reset=%b, flush=%b,
insert=%b, remove=%b, datain=%h, dataout=%h, empty=%b, full=%b
FAIL",clk_in, clk_out,rst,flush,insert,remove,data_in,data_out,empty,full);
end
end

endmodule
```

SIMULATION RESULTS

```
        Chronologic VCS simulator copyright 1991-2014
Contains Synopsys proprietary information.
Compiler version I-2014.03-2; Runtime version I-2014.03-2; Dec 1 11:14 2016
VCD+ Writer I-2014.03-2 Copyright (c) 1991-2014 by Synopsys Inc.
          4
clk_in=0,clk_out=1,reset=0,flush=0,insert=0,remove=0,datain=xxxxxxxx,dataout=00000000,empty=1,
full=0 PASS
         12
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=12153524,dataout=00000000,empty=1,
full=0 PASS
         20
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=12153524,dataout=00000000,empty=1,
full=0 PASS
         28
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=12153524,dataout=00000000,empty=1,
full=0 PASS
         36
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=b1f05663,dataout=00000000,empty=1,f
ull=0 PASS
         44
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=b1f05663,dataout=12153524,empty=0,f
ull=0 PASS
         52
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=b1f05663,dataout=12153524,empty=0,f
ull=0 PASS
         60
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=b2c28465,dataout=12153524,empty=0,
full=0 PASS
         68
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=b2c28465,dataout=b1f05663,empty=0,f
ull=0 PASS
         76
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=b2c28465,dataout=b1f05663,empty=0,f
ull=0 PASS
         84
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=06d7cd0d,dataout=b1f05663,empty=0,f
ull=0 PASS
         92
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=06d7cd0d,dataout=b1f05663,empty=1,f
ull=0 PASS
        100
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=06d7cd0d,dataout=b1f05663,empty=1,f
ull=0 PASS
        108
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=76d457ed,dataout=b1f05663,empty=1,f
ull=0 PASS
        116
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=76d457ed,dataout=b1f05663,empty=1,f
ull=0 PASS
```

```
            124
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=76d457ed,dataout=b1f05663,empty=1,f
ull=0 PASS
            132
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e33724c6,dataout=b1f05663,empty=1,f
ull=0 PASS
            140
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e33724c6,dataout=b1f05663,empty=1,f
ull=0 PASS
            148
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e33724c6,dataout=b1f05663,empty=1,f
ull=0 PASS
            156
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=72aff7e5,dataout=b1f05663,empty=1,f
ull=0 PASS
            164
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=72aff7e5,dataout=b1f05663,empty=0,f
ull=0 PASS
            172
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=72aff7e5,dataout=b1f05663,empty=0,f
ull=0 PASS
            180
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=47ecdb8f,dataout=b1f05663,empty=0,f
ull=0 PASS
            188
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=47ecdb8f,dataout=b1f05663,empty=0,f
ull=0 PASS
            196
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=47ecdb8f,dataout=b1f05663,empty=0,f
ull=0 PASS
            204
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=f4007ae8,dataout=b1f05663,empty=0,f
ull=0 PASS
            212
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=f4007ae8,dataout=b1f05663,empty=0,f
ull=0 PASS
            220
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=f4007ae8,dataout=b1f05663,empty=0,f
ull=0 PASS
            228
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=de8e28bd,dataout=b1f05663,empty=0,f
ull=0 PASS
            236
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=de8e28bd,dataout=b1f05663,empty=0,f
ull=0 PASS
            244
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=de8e28bd,dataout=b1f05663,empty=0,f
ull=0 PASS
            252
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=b1ef6263,dataout=b1f05663,empty=0,f
ull=0 PASS
```

```
            260
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=b1ef6263,dataout=e33724c6,empty=0,f
ull=0 PASS
            268
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=b1ef6263,dataout=e33724c6,empty=0,f
ull=0 PASS
            276
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=10642120,dataout=e33724c6,empty=0,
full=0 PASS
            284
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=10642120,dataout=e33724c6,empty=0,
full=0 PASS
            292
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=10642120,dataout=e33724c6,empty=0,
full=0 PASS
            300
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=cb203e96,dataout=e33724c6,empty=0,f
ull=0 PASS
            308
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=cb203e96,dataout=72aff7e5,empty=0,f
ull=0 PASS
            316
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=cb203e96,dataout=72aff7e5,empty=0,f
ull=0 PASS
            324
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=a9a7d653,dataout=72aff7e5,empty=0,f
ull=0 PASS
            332
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=a9a7d653,dataout=f4007ae8,empty=0,f
ull=0 PASS
            340
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=a9a7d653,dataout=f4007ae8,empty=0,f
ull=0 PASS
            348
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=81174a02,dataout=f4007ae8,empty=0,f
ull=0 PASS
            356
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=81174a02,dataout=de8e28bd,empty=0,f
ull=0 PASS
            364
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=81174a02,dataout=de8e28bd,empty=0,f
ull=0 PASS
            372
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e7c572cf,dataout=de8e28bd,empty=0,f
ull=0 PASS
            380
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e7c572cf,dataout=cb203e96,empty=0,f
ull=0 PASS
            388
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e7c572cf,dataout=cb203e96,empty=0,f
ull=0 PASS
```

```
            396
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e5730aca,dataout=cb203e96,empty=0,f
ull=0 PASS
            404
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e5730aca,dataout=cb203e96,empty=0,f
ull=0 PASS
            412
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e5730aca,dataout=cb203e96,empty=0,f
ull=0 PASS
            420
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=452e618a,dataout=cb203e96,empty=0,f
ull=0 PASS
            428
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=452e618a,dataout=cb203e96,empty=0,f
ull=0 PASS
            436
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=452e618a,dataout=cb203e96,empty=0,f
ull=0 PASS
            444
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=3c20f378,dataout=cb203e96,empty=0,f
ull=0 PASS
            452
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=3c20f378,dataout=cb203e96,empty=0,f
ull=0 PASS
            460
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=3c20f378,dataout=cb203e96,empty=0,f
ull=0 PASS
            468
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=5b0265b6,dataout=cb203e96,empty=0,
full=0 PASS
            476
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=5b0265b6,dataout=a9a7d653,empty=0,
full=0 PASS
            484
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=5b0265b6,dataout=a9a7d653,empty=0,
full=0 PASS
            492
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=de7502bc,dataout=a9a7d653,empty=0,f
ull=0 PASS
            500
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=de7502bc,dataout=a9a7d653,empty=0,f
ull=0 PASS
            508
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=de7502bc,dataout=a9a7d653,empty=0,f
ull=0 PASS
            516
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=b897be71,dataout=a9a7d653,empty=0,f
ull=0 PASS
            524
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=b897be71,dataout=e7c572cf,empty=0,f
ull=0 PASS
```

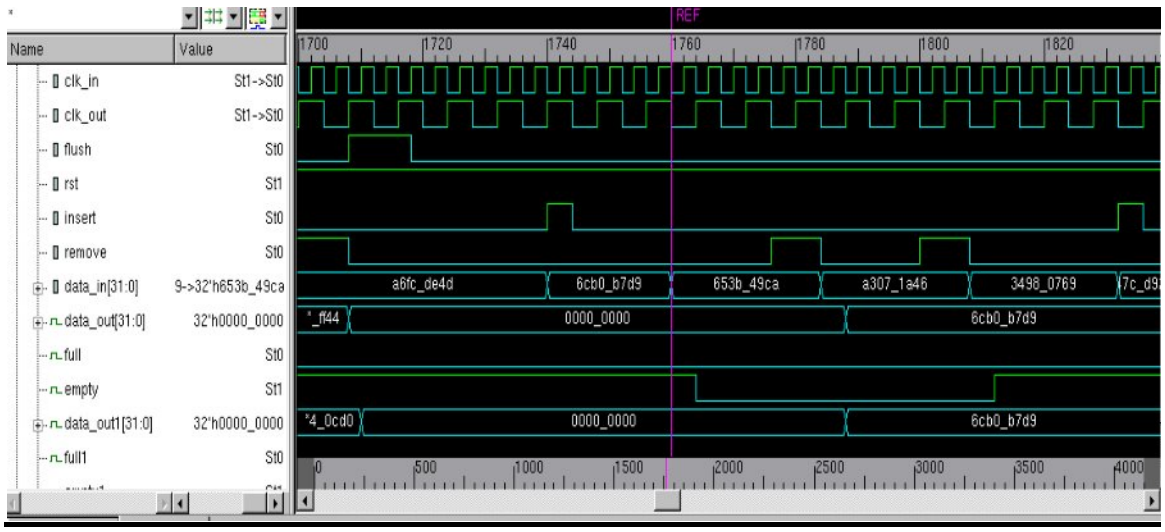
```
                532
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=b897be71,dataout=e7c572cf,empty=0,f
ull=0 PASS
                540
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=9dcc603b,dataout=e7c572cf,empty=0,f
ull=0 PASS
                548
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=9dcc603b,dataout=452e618a,empty=0,f
ull=0 PASS
                556
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=9dcc603b,dataout=452e618a,empty=0,f
ull=0 PASS
                564
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=0aaa4b15,dataout=452e618a,empty=0,f
ull=0 PASS
                572
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=0aaa4b15,dataout=452e618a,empty=0,f
ull=0 PASS
                580
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=0aaa4b15,dataout=452e618a,empty=0,f
ull=0 PASS
                588
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=31230762,dataout=452e618a,empty=0,
full=0 PASS
                596
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=31230762,dataout=3c20f378,empty=0,f
ull=0 PASS
                604
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=31230762,dataout=3c20f378,empty=0,f
ull=0 PASS
                612
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=47b9a18f,dataout=3c20f378,empty=0,f
ull=0 PASS
                620
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=47b9a18f,dataout=b897be71,empty=0,f
ull=0 PASS
                628
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=47b9a18f,dataout=b897be71,empty=0,f
ull=0 PASS
                636
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=cfc4569f,dataout=b897be71,empty=0,f
ull=0 PASS
                644
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=cfc4569f,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                652
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=cfc4569f,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                660
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=44de3789,dataout=0aaa4b15,empty=0,f
ull=0 PASS
```

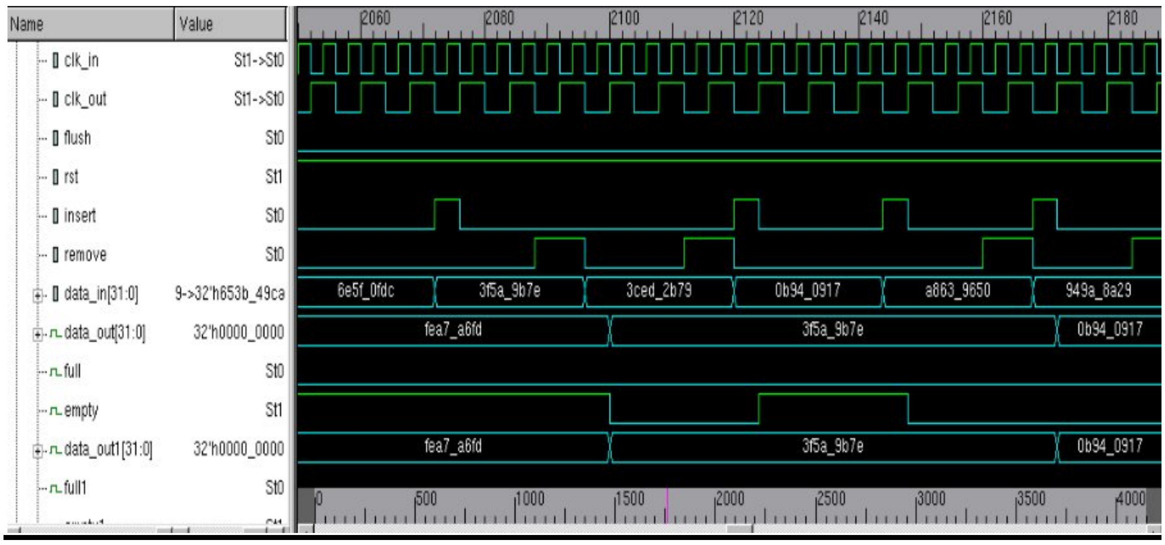
```
                 668
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=44de3789,dataout=0aaa4b15,empty=1,f
ull=0 PASS
                 676
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=44de3789,dataout=0aaa4b15,empty=1,f
ull=0 PASS
                 684
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=ebfec0d7,dataout=0aaa4b15,empty=1,f
ull=0 PASS
                 692
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=ebfec0d7,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                 700
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=ebfec0d7,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                 708
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=061d7f0c,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                 716
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=061d7f0c,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                 724
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=061d7f0c,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                 732
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=bb825a77,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                 740
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=bb825a77,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                 748
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=bb825a77,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                 756
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=bf05007e,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                 764
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=bf05007e,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                 772
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=bf05007e,dataout=0aaa4b15,empty=0,f
ull=0 PASS
                 780
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=0fd28f1f,dataout=0aaa4b15,empty=0,fu
ll=0 PASS
                 788
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=0fd28f1f,dataout=44de3789,empty=0,f
ull=0 PASS
                 796
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=0fd28f1f,dataout=44de3789,empty=0,f
ull=0 PASS
```

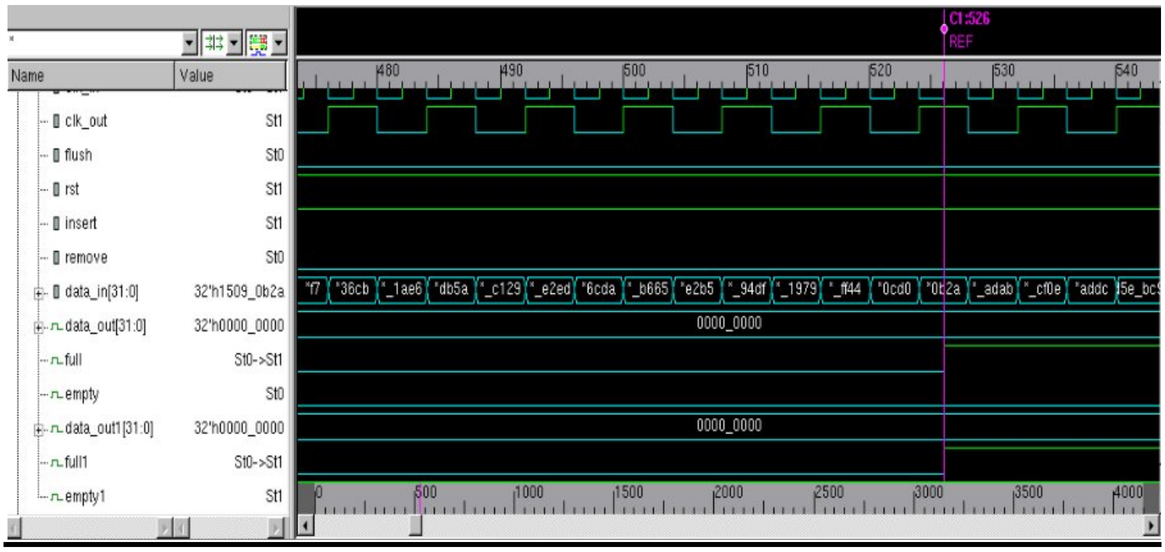
```
                804
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=bc148878,dataout=44de3789,empty=0,
full=0 PASS
                812
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=bc148878,dataout=ebfec0d7,empty=0,f
ull=0 PASS
                820
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=bc148878,dataout=ebfec0d7,empty=0,f
ull=0 PASS
                828
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=9ff2ae3f,dataout=ebfec0d7,empty=0,ful
l=0 PASS
                836
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=9ff2ae3f,dataout=bb825a77,empty=0,fu
ll=0 PASS
                844
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=9ff2ae3f,dataout=bb825a77,empty=0,fu
ll=0 PASS
                852
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=c33f3886,dataout=bb825a77,empty=0,f
ull=0 PASS
                860
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=c33f3886,dataout=bb825a77,empty=0,f
ull=0 PASS
                868
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=c33f3886,dataout=bb825a77,empty=0,f
ull=0 PASS
                876
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=7d3599fa,dataout=bb825a77,empty=0,f
ull=0 PASS
                884
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=7d3599fa,dataout=bb825a77,empty=0,f
ull=0 PASS
                892
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=7d3599fa,dataout=bb825a77,empty=0,f
ull=0 PASS
                900
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=d18bb4a3,dataout=bb825a77,empty=0,
full=0 PASS
                908
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=d18bb4a3,dataout=bf05007e,empty=0,f
ull=0 PASS
                916
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=d18bb4a3,dataout=bf05007e,empty=0,f
ull=0 PASS
                924
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=afd8565f,dataout=bf05007e,empty=0,f
ull=0 PASS
                932
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=afd8565f,dataout=0fd28f1f,empty=0,fu
ll=0 PASS
```

```
                 940
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=1,datain=afd8565f,dataout=0fd28f1f,empty=0,fu
ll=0 PASS
                 948
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e59b36cb,dataout=0fd28f1f,empty=0,f
ull=0 PASS
                 956
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e59b36cb,dataout=bc148878,empty=0,f
ull=0 PASS
                 964
clk_in=0,clk_out=1,reset=1,flush=0,insert=0,remove=0,datain=e59b36cb,dataout=bc148878,empty=0,f
ull=0 PASS
                 972
clk_in=0,clk_out=1,reset=0,flush=0,insert=0,remove=0,datain=e59b36cb,dataout=00000000,empty=1,
full=0 PASS
$finish called from file "fifo_fixture.v", line 82.
$finish at simulation time                  978
           V C S S i m u l a t i o n R e p o r t
Time: 978
CPU Time:      0.340 seconds;       Data structure size:   0.0Mb
Thu Dec  1 11:14:08 2016
```

SIMULATION WAVEFORMS







SYNTHESIS

a. Synthesis script file:

```
#Scripting

sh mkdir WORK
define_design_lib WORK -path ./WORK
analyze -library WORK -format verilog {fifo.v}
elaborate fifo -architecture verilog -library WORK

#set the current design
set current_design fifo

#Link the design
link

#create_clock "clk_in" -period 2 -name "clk_in"
#set_input_delay 2.0 -clock clk_in [all_inputs]
#create clockand constrain the design
create_clock "clk_in" -period 2 -name  "clk_in"
set_input_delay -clock clk_in -max -rise 0.2 "insert"
set_input_delay -clock clk_in -min -rise 0.02 "insert"
set_input_delay -clock clk_in -max -rise 0.2 "flush"
set_input_delay -clock clk_in -min -rise 0.02 "flush"
set_input_delay -clock clk_in -max -rise 0.2 "data_in"
set_input_delay -clock clk_in -min -rise 0.02 "data_in"
set_output_delay -clock clk_in -max -rise 0.2 "full"
set_output_delay -clock clk_in -min -rise 0.02 "full"
create_clock "clk_out" -period 4 -name "clk_out"
set_input_delay -clock clk_out -max -rise 0.1 "remove"
set_input_delay -clock clk_out -min -rise 0.01 "remove"
set_output_delay -clock clk_out -max -rise 0.1 "empty"
set_output_delay -clock clk_out -min -rise 0.01 "empty"
set_output_delay -clock clk_out -max -rise 0.1 "data_out"
set_output_delay -clock clk_out  -min -rise 0.01 "data_out"

set_dont_touch_network "clk_in"
set_dont_touch_network "clk_out"
```

```
set_max_area 0
set_false_path -from [get_clocks {clk_in}] -to [get_clocks {clk_out}]

#Set operating conditions
set_operating_conditions -library "saed90nm_typ" "TYPICAL"
#Synthesize and generate report
compile -map_effort high -boundary_optimization
report_attribute >  report1
report_area >  report2
report_constraints -all_violators > report3
report_timing -path full -delay max -max_paths 1 -nworst 1  > report4
```

b. Synthesis Reports:

I. Report 1: Area

```
****************************************
Report : area
Design : fifo
Version: I-2013.12-SP5-4
Date   : Thu Dec  1 02:52:25 2016
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    saed90nm_typ (File:
/netdisk/tmp/saed/SAED90_EDK/SAED_EDK90nm/Digital_Standard_cell_Library/s
ynopsys/models/saed90nm_typ.db)

Number of ports:                       72
Number of nets:                       153
Number of cells:                        4
Number of combinational cells:          0
Number of sequential cells:             0
Number of macros/black boxes:           0
Number of buf/inv:                      0
Number of references:                   4

Combinational area:          128527.259138
Buf/Inv area:                 12362.342672
Noncombinational area:        95329.384798
Macro/Black Box area:             0.000000
Net Interconnect area:        23501.553884
```

```
Total cell area:              223856.643936
Total area:                  247358.197820
```

II. Report 2: Constraints

```
****************************************
Report : constraint
        -all_violators
Design : fifo
Version: I-2013.12-SP5-4
Date   : Thu Dec  1 02:52:27 2016
****************************************

   max_area

                          Required        Actual
   Design                     Area          Area          Slack
   -----------------------------------------------------------------
   fifo                       0.00      247358.20     -247358.20
                                                      (VIOLATED)
```

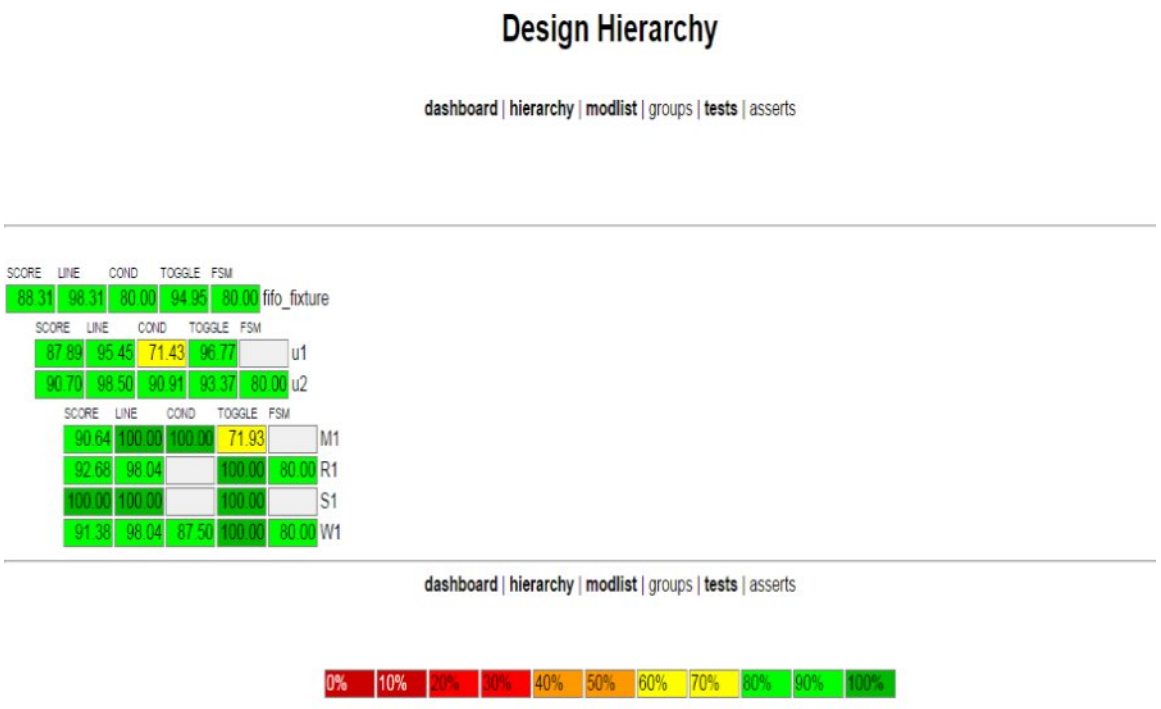
III. Report 3: Timing

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : fifo
Version: I-2013.12-SP5-4
Date   : Thu Dec  1 02:52:27 2016
****************************************

Operating Conditions: TYPICAL   Library: saed90nm_typ
Wire Load Model Mode: enclosed

  Startpoint: S1/read_pointer_syn_reg[3]
              (rising edge-triggered flip-flop clocked by clk_in)
  Endpoint: W1/write_pointer_reg[3]
            (rising edge-triggered flip-flop clocked by clk_in)
  Path Group: clk_in
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  fifo               280000                saed90nm_typ
  write_P7_C6        8000                  saed90nm_typ

  Point                                                   Incr       Path
  --------------------------------------------------------------------------
  clock clk_in (rise edge)                                0.00       0.00
```

```
S1/read_pointer_syn_reg[3]/Q (DFFARX1)          0.17      0.17 r
S1/read_pointer_syn[3] (Syncronizer_P7)         0.00      0.17 r
W1/read_pointer_syn[3] (write_P7_C6)            0.00      0.17 r
W1/U57/Q (XOR2X1)                               0.94      1.11 r
W1/U52/QN (NOR4X0)                              0.12      1.23 f
W1/U9/QN (NAND4X0)                              0.10      1.33 r
W1/U8/Q (OR2X1)                                 0.12      1.45 r
W1/U32/Q (AND3X1)                               0.13      1.57 r
W1/U47/Q (OA21X1)                               0.15      1.72 r
W1/U28/Q (AO22X1)                               0.14      1.86 r
W1/write_pointer_reg[3]/D (DFFARX1)             0.03      1.89 r
data arrival time                                         1.89

clock clk_in (rise edge)                        2.00      2.00
clock network delay (ideal)                     0.00      2.00
W1/write_pointer_reg[3]/CLK (DFFARX1)           0.00      2.00 r
library setup time                             -0.09      1.91
data required time                                        1.91
--------------------------------------------------------------------------
data required time                                        1.91
data arrival time                                        -1.89
--------------------------------------------------------------------------
slack (MET)                                               0.02


Startpoint: R1/read_pointer_reg[0]
          (rising edge-triggered flip-flop clocked by clk_out)
Endpoint: R1/read_pointer_reg[0]
         (rising edge-triggered flip-flop clocked by clk_out)
Path Group: clk_out
Path Type: max

Des/Clust/Port     Wire Load Model       Library
------------------------------------------------
fifo              280000              saed90nm_typ
read_P7_C6         8000                saed90nm_typ

Point                                           Incr      Path
--------------------------------------------------------------------------
clock clk_out (rise edge)                       0.00      0.00
clock network delay (ideal)                     0.00      0.00
R1/read_pointer_reg[0]/CLK (DFFARX1)            0.00      0.00 r
R1/read_pointer_reg[0]/Q (DFFARX1)              0.19      0.19 r
R1/U44/Q (XOR2X1)                               1.56      1.75 r
R1/U40/QN (NOR4X0)                              0.11      1.86 f
R1/U39/QN (NAND4X0)                             0.11      1.97 r
R1/U34/ZN (INVX0)                               0.09      2.06 f
R1/U29/Q (AO221X1)                              0.16      2.21 f
R1/U26/QN (NOR3X0)                              0.13      2.34 r
R1/U25/Q (AO21X1)                               0.16      2.50 r
R1/U24/QN (NOR3X0)                              0.28      2.78 f
R1/U14/Q (OA21X1)                               0.16      2.94 f
R1/U5/Q (AO22X1)                                0.14      3.08 f
```

```
clock clk_out (rise edge)                      4.00     4.00
clock network delay (ideal)                    0.00     4.00
R1/read_pointer_reg[0]/CLK (DFFARX1)           0.00     4.00 r
library setup time                            -0.06     3.94
data required time                                      3.94
--------------------------------------------------------------------------
data required time                                      3.94
data arrival time                                      -3.12
--------------------------------------------------------------------------
slack (MET)                                             0.83
```

Code coverage report



Design Hierarchy

dashboard | hierarchy | modlist | groups | tests | asserts

SCORE	LINE	COND	TOGGLE	FSM	
88.31	98.31	80.00	94.95	80.00	fifo_fixture

SCORE	LINE	COND	TOGGLE	FSM	
87.89	95.45	71.43	96.77		u1
90.70	98.50	90.91	93.37	80.00	u2

SCORE	LINE	COND	TOGGLE	FSM	
90.64	100.00	100.00	71.93		M1
92.68	98.04		100.00	80.00	R1
100.00	100.00		100.00		S1
91.38	98.04	87.50	100.00	80.00	W1

dashboard | hierarchy | modlist | groups | tests | asserts

0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

Trial 2 Synthesis report4

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : fifo
Version: I-2013.12-SP5-4
Date   : Sun Nov 20 21:56:12 2016
****************************************

 # A fanout number of 1000 was used for high fanout net computations.

Operating Conditions: TYPICAL   Library: saed90nm_typ
Wire Load Model Mode: enclosed

  Startpoint: W1/write_address_reg[0]
              (rising edge-triggered flip-flop clocked by clk_in)
  Endpoint: M1/mem_reg[90][2]
            (rising edge-triggered flip-flop clocked by clk_in)
  Path Group: clk_in
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  fifo               540000                saed90nm_typ
  memory_M32_N128_C6 280000                saed90nm_typ

  Point                                                  Incr       Path
  --------------------------------------------------------------------------
  clock clk_in (rise edge)                               0.00       0.00
  clock network delay (ideal)                            0.00       0.00
  W1/write_address_reg[0]/CLK (DFFARX1)                  0.00 #     0.00 r
  W1/write_address_reg[0]/Q (DFFARX1)                    0.18       0.18 r
  W1/write_address[0] (write_P7_C6)                      0.00       0.18 r
  M1/write_address[0] (memory_M32_N128_C6)               0.00       0.18 r
  M1/U5243/Z (DELLN1X2)                                  0.73       0.91 r
  M1/U8200/Z (DELLN1X2)                                  0.72       1.63 r
  M1/U5440/Z (DELLN1X2)                                  0.77       2.40 r
  M1/U4906/Z (DELLN1X2)                                  0.69       3.09 r
  M1/U6989/Q (MUX41X1)                                   1.93       5.01 r
  M1/U134/Q (MUX41X2)                                    0.58       5.60 r
  M1/U7008/Q (MUX41X1)                                   0.56       6.15 r
  M1/U4563/Q (MUX21X2)                                   0.61       6.77 r
  M1/U5654/Q (AO22X2)                                    0.60       7.37 r
  M1/U5485/Z (DELLN1X2)                                  0.77       8.14 r
  M1/U3119/Q (AO22X1)                                    0.82       8.95
```

```
data arrival time                                  9.29

clock clk_in (rise edge)                         2.00       2.00
clock network delay (ideal)                      0.00       2.00
M1/mem_reg[90][2]/CLK (DFFARX1)        0.00       2.00 r
library setup time                               -0.09       1.91
data required time                                           1.91
-----------------------------------------------------------------------
data required time                                           1.91
data arrival time                                           -9.29
-----------------------------------------------------------------------
slack (VIOLATED)                                            -7.37



Startpoint: R1/read_address_reg[0]
            (rising edge-triggered flip-flop clocked by clk_out)
Endpoint: M1/data_out_reg[14]
          (rising edge-triggered flip-flop clocked by clk_out)
Path Group: clk_out
Path Type: max

Des/Clust/Port     Wire Load Model       Library
------------------------------------------------
fifo               540000                saed90nm_typ
memory_M32_N128_C6 280000               saed90nm_typ

Point                                                    Incr       Path
-----------------------------------------------------------------------
clock clk_out (rise edge)                                 0.00       0.00
clock network delay (ideal)                               0.00       0.00
R1/read_address_reg[0]/CLK (DFFARX1)             0.00       0.00 r
R1/read_address_reg[0]/Q (DFFARX1)               0.17       0.17 r
R1/read_address[0] (read_P7_C6)                  0.00       0.17 r
M1/read_address[0] (memory_M32_N128_C6)        0.00       0.17 r
M1/U3348/Z (DELLN1X2)                                0.68       0.85 r
M1/U1949/Z (DELLN1X2)                                0.72       1.57 r
M1/U48/ZN (INVX0)                                       0.96       2.53 f
M1/U5349/ZN (INVX0)                                   0.61       3.14 r
M1/U8775/Q (MUX41X1)                                 1.76       4.91 r
M1/U8777/Q (MUX41X1)                                 0.54       5.44 r
M1/U8787/Q (MUX41X1)                                 0.58       6.02 r
M1/U1002/Q (MUX21X2)                                 0.61       6.63 r
M1/U4443/Q (AND2X1)                                    0.44       7.07 r
M1/data_out_reg[14]/D (DFFARX1)                   0.33       7.40 r
data arrival time                                                    7.40

clock clk_out (rise edge)                                4.00       4.00
clock network delay (ideal)                              0.00       4.00
M1/data_out_reg[14]/CLK (DFFARX1)                0.00       4.00 r
```

```
 -------------------------------------------------------------------------
 data required time                                   3.92
 data arrival time                                   -7.40
 -------------------------------------------------------------------------
 slack (VIOLATED)                                          -3.48
```

Sequential memory module previously used for getting above report

```
//Sequential memory block -first trial

module memory#(parameter M=32,N=128,C=6)(input
clk_in,clk_out,rst,flush,write_enable,read_enable,input[M-1:0] data_in,input [C:0]
write_address,read_address,output reg [M-1:0] data_out);
reg [M-1:0] mem[N-1:0];
integer i;

always @ (posedge clk_in or negedge rst)  //write block
begin
    if (!rst)
    begin
        for (i=0; i < 128; i=i+1)
        begin
            mem[i] <= 32'h0;
        end
    end

    else if(flush)
      begin
        for(i=0; i < N; i=i+1)
          mem[i] <= 32'h0;
      end

    else if (write_enable)
    begin
        mem[write_address] <= data_in;
    end


    else
    begin
        mem[write_address] <= mem[write_address];
    end
end

always @(posedge clk_out or negedge rst) //read block
begin
    if(!rst)
    begin
       data_out<=32'h0;
    end
```

```
       data_out<=mem[read_address];
    end
    else
    begin
        data_out<=32'h0;
    end
end
endmodule
```