**Code:**

* **Design Module**
* **Parameter module**

`define Bus\_Length 32

* **Main Module**

// Code your design here

`include"parameter.sv"

//replaced ports with parameterized initiation

// module reg\_file(input clk, //ports

// input clr,

// input regwrite, //operation

// input [4:0] writereg,

// input [4:0] readreg1,

// input [4:0] readreg2,

// input [31:0] data,

// output [31:0] read1,

// output[31: 0] read2);

// reg [31 :0] out1;

// reg [31 :0] out2;

// reg [31:0] reg\_device [32:0];

module reg\_file(input clk, //ports

input clr,

input regwrite, // operation

input [$clog2(`Bus\_Length) -1 :0] writereg,

input [$clog2(`Bus\_Length) -1 :0] readreg1,

input [$clog2(`Bus\_Length) -1 :0] readreg2,

input [(`Bus\_Length) -1 :0] data,

output [(`Bus\_Length) -1 :0] read1,

output[(`Bus\_Length) -1 : 0] read2);

reg [(`Bus\_Length) -1 :0] out1;

reg [(`Bus\_Length) -1 :0] out2;

reg [(`Bus\_Length) -1 :0] reg\_device [(`Bus\_Length) : 0 ]; // internal memory

always@(posedge clk)

begin

if(clr)

begin

out1 <= 0;

out2 <= 0;

for(int i = 0; i <= (`Bus\_Length - 1); i++ )

begin

reg\_device[i] <= 0;

end

end

else

begin

out1 <= reg\_device[readreg1];

out2 <= reg\_device[readreg2];

end

end

always@(negedge clk)

begin

if(clr)

begin

out1 <= 0;

out2 <= 0;

for(int i = 0; i <= (`Bus\_Length - 1); i++ )

begin

reg\_device[i] <= 0;

end

end

else

begin

out1 <= reg\_device[readreg1];

out2 <= reg\_device[readreg2];

if(regwrite)

begin

reg\_device [writereg ] <= data;

end

end

end

assign read1 = out1;

assign read2 = out2;

endmodule

* **TestBench**

// Code your testbench here

// or browse Examples

module tb();

bit clk,clr,regwrite;

reg [$clog2(`Bus\_Length) -1 :0] writereg; //location for write

reg [$clog2(`Bus\_Length) -1 :0] readreg1; // location for read

reg [$clog2(`Bus\_Length) -1 :0] readreg2;

reg [(`Bus\_Length) -1 :0] data;

wire [(`Bus\_Length) -1 :0] read1;

wire [(`Bus\_Length) -1 : 0] read2;

reg\_file dut(clk,clr,regwrite,writereg,readreg1,readreg2,data,read1,read2);

always #5 clk = ~clk;

initial

begin

clk <= 1;

clr <= 0;

regwrite <= 1;

writereg <= 32'b1001;

data <= 32'b1111;

#10

readreg1 <= 32'b1001;

regwrite <= 0;

#5;

end

initial

begin

#20

regwrite <= 1;

writereg <= 32'b010001;

data <= 32'b100001;

#15

regwrite <= 0;

readreg1 <= 32'b010001;

#10

clr <= 1;

end

//for eda playground waveform

// initial

// begin

// $dumpfile("dump.vcd");

// $dumpvars;

// #500

// $finish;

// end

endmodule

**Output:**

Diagram

Description automatically generated