Some sample submission for “RAM and Register File”

Verilog file and Testbench

Sample format for submission:

Student Name= __________ Roll No.= _________________

Verilog file:

Test bench

Student Name: Saraswatula Phani Sai Pranav **Roll No**:180101070

Verilog File: RAM module

```
module ram_module(input clk,input [1:0]data_in,input wr_en,input rd_en,input [3:0]address,output reg
[1:0]data_out

   );

reg [1:0]mem[15:0];integer i;

initial begin

for(i=0;i<16;i=i+1)begin

   mem[i]<=2'b00;

end

end


always @(posedge clk)

begin

if(wr_en)

begin

   mem[address]<=data_in;

end

end

always #0.001

begin

if(rd_en)

begin
```

```
data_out<=mem[address];
end
end
endmodule
```

Testbench:

```
module ram_module_test;

   // Inputs
   reg clk;
   reg [1:0] data_in;
   reg wr_en;
   reg rd_en;
   reg [3:0] address;

   // Outputs
   wire [1:0] data_out;

   // Instantiate the Unit Under Test (UUT)
   ram_module uut (
      .clk(clk),
      .data_in(data_in),
      .wr_en(wr_en),
      .rd_en(rd_en),
      .address(address),
      .data_out(data_out)
   );
initial begin
clk=0;
forever #20 clk=~clk;
```

```
end
  initial begin
    // Initialize Inputs
    data_in = 2'b10;
    wr_en = 1;
    rd_en = 1;
    address = 4'b0010;

    // Wait 100 ns for global reset to finish
    #40;

    // Add stimulus here
    data_in = 2'b11;
    wr_en = 1;
    rd_en = 1;
    address = 4'b0110;

    // Wait 100 ns for global reset to finish
    #40;
        data_in = 2'b10;
    wr_en = 0;
    rd_en = 1;
    address = 4'b0010;

    // Wait 100 ns for global reset to finish
    #40;
        data_in = 2'b01;
    wr_en = 0;
    rd_en = 1;
```

```
        address = 4'b0110;

        #5;

        rd_en=1;

        wr_en=0;

        data_in=2'b00;

        address=4'b0010;

    end

endmodule
```

Verilog File: Register File

```
module register_file(input clk,input reset,input [2:0]read_select,input [2:0]write_select,input
write_enable,input [7:0]data_in,output reg [7:0]data_out

    );

reg [7:0]mem[7:0];

initial begin

mem[0]=0;mem[1]=0;mem[2]=0;mem[3]=0;

mem[4]=0;mem[5]=0;mem[6]=0;mem[7]=0;

end

always @(posedge clk)

begin

if(write_enable)

begin

mem[write_select]<=data_in;

end

end

always #0.001

begin

data_out<= mem[read_select];

end

always @(posedge reset)
```

```
begin
mem[0]=0;mem[1]=0;mem[2]=0;mem[3]=0;
mem[4]=0;mem[5]=0;mem[6]=0;mem[7]=0;
end
endmodule
```

Test Bench:

```
module register_file_test;

  // Inputs
  reg clk;
  reg reset;
  reg [2:0]read_select;
  reg [2:0]write_select;
  reg write_enable;
  reg [7:0] data_in;

  // Outputs
  wire [7:0]data_out;

  // Instantiate the Unit Under Test (UUT)
  register_file uut (
    .clk(clk),
    .reset(reset),
    .read_select(read_select),
    .write_select(write_select),
    .write_enable(write_enable),
    .data_in(data_in),
    .data_out(data_out)
  );
```

```
initial begin
clk=0;
forever #20 clk=~clk;
end
 initial begin
     // Initialize Inputs
     reset=1;
     read_select = 3'b010;
     write_select = 3'b010;
     write_enable = 1;
     data_in = 8'b00101000;
     #40;
     reset=0;
     read_select = 3'b010;
     write_select = 3'b010;
     write_enable = 1;
     data_in = 8'b01101000;
     #40;

     // Add stimulus here
     read_select = 3'b011;
     write_select = 3'b111;
     write_enable = 1;
     data_in = 8'b10101001;
     #40;
     read_select = 3'b101;
     write_select = 3'b101;
     write_enable = 1;
     data_in = 8'b00101111;
```

```
        #40;

        read_select = 3'b111;

        write_select = 3'b111;

        write_enable = 1;

        data_in = 8'b01101011;

        #40;

        read_select = 3'b000;

        write_select = 3'b010;

        write_enable = 1;

        data_in = 8'b00101110;

        #40;

        reset=1;

        read_select = 3'b000;

        write_select = 3'b000;

        write_enable = 0;

        data_in = 8'b00101000;

        #5;

        read_select = 3'b111;

        write_select = 3'b000;

        write_enable = 0;

        data_in = 8'b00101000;

        #5;

        read_select = 3'b111;

        write_select = 3'b111;

        write_enable = 1;

        data_in = 8'b00101011;

    end


endmodule
```

Student Name: Kartikeya Saxena   **Roll No**:180101034

Verilog File: RAM module

```
`timescale 1ns / 1ns

module RAM(clk, data_in, wr_en, rd_en, address, data_out);

input clk;

input [15:0] data_in;

input wr_en;

input rd_en;

input [3:0] address;

output reg [15:0] data_out;



reg [15:0] memory [15:0];

integer i, j;

initial
```

```
begin
for(i = 0; i < 16; i = i + 1)
memory[i] <= 16'b0;
end


always @(posedge clk)
begin
if(wr_en)
memory[address] <= data_in;
end


always @(*)
begin
if(rd_en)
data_out <= memory[address];
end
Endmodule
```

Testbench:

```
`timescale 1ns / 1ns
module RAM_test_bench();
reg clk;
reg [15:0] data_in;
reg wr_en;
reg rd_en;
reg [3:0] address;
wire [15:0] data_out;
```

```
RAM x(.clk(clk), .data_in(data_in), .wr_en(wr_en), .rd_en(rd_en), .address(address), .data_out(data_out));

initial
begin
$dumpfile("output.vcd");
$dumpvars(0, x);
clk = 1'b0;
forever
#5 clk = ~clk;
end

initial
begin
$monitor("time = %3d, data_in = %d, data_out = %d, address = %d\n", $time, data_in, data_out, address);
end

initial
begin
#2
#10 wr_en = 1; address = 2; data_in = 5;
#1 wr_en = 0; rd_en = 1;
#1 wr_en = 1; rd_en = 0;
#10 wr_en = 0; rd_en = 1; address = 2;
#10 address = 5;
#10 rd_en = 0; wr_en = 1; data_in = 2;
```

```
#10 wr_en = 0; rd_en = 1; address = 0;
#10 address = 2;
#10 address = 5;
#10 $finish;
end
endmodule
```

Verilog File: RF module

```
`timescale 1ns / 1ns
module RF(clk, reset, rd_sel, wr_sel, wr_en, data_in, data_out);
input clk;
input reset;
input [3:0] rd_sel;
input [3:0] wr_sel;
input wr_en;
input [7:0] data_in;
output reg [7:0] data_out;
reg [7:0] r [7:0];
integer i;


always @(posedge clk)
begin
if(reset == 1'b0)
begin
if(wr_en)
r[wr_sel] <= data_in;
end
end
```

```
always @(*)
begin
data_out <= r[rd_sel];
end


always @(posedge reset)
begin
for(i = 0; i < 8; i = i + 1)
begin
r[i] <= 8'b0;
end
end


endmodule
```

Testbench:

```
`timescale 1ns / 1ns
module RF_test_bench();
reg clk;
reg reset;
reg [3:0] rd_sel;
reg [3:0] wr_sel;
reg wr_en;
reg [7:0] data_in;
wire [7:0] data_out;
```

```
RF x(.clk(clk), .reset(reset), .rd_sel(rd_sel), .wr_sel(wr_sel), .wr_en(wr_en), .data_in(data_in),
.data_out(data_out));

initial

begin

$dumpfile("output.vcd");

$dumpvars(0, x);

clk = 1'b0;

reset = 1'b0;

forever

#5 clk = ~clk;

end

initial

begin

$monitor("time = %3d, data_in = %d, data_out = %d, rd_sel = %d, wr_sel = %d\n", $time, data_in, data_out,
rd_sel, wr_sel);

end

initial

begin

#2

#10 reset = 1;

#10 rd_sel = 4;

#10 wr_en = 1; wr_sel = 5; data_in = 9;

#10 rd_sel = 5;
```

```
#10 reset = 0;

#10 wr_sel = 1; data_in = 2;

#10 rd_sel = 2; wr_sel = 2; data_in = 6;

#10 rd_sel = 2;

#10 wr_sel = 5; data_in = 10;

#10 wr_en = 0;

#10 rd_sel = 5;

#10 $finish;

end

endmodule
```