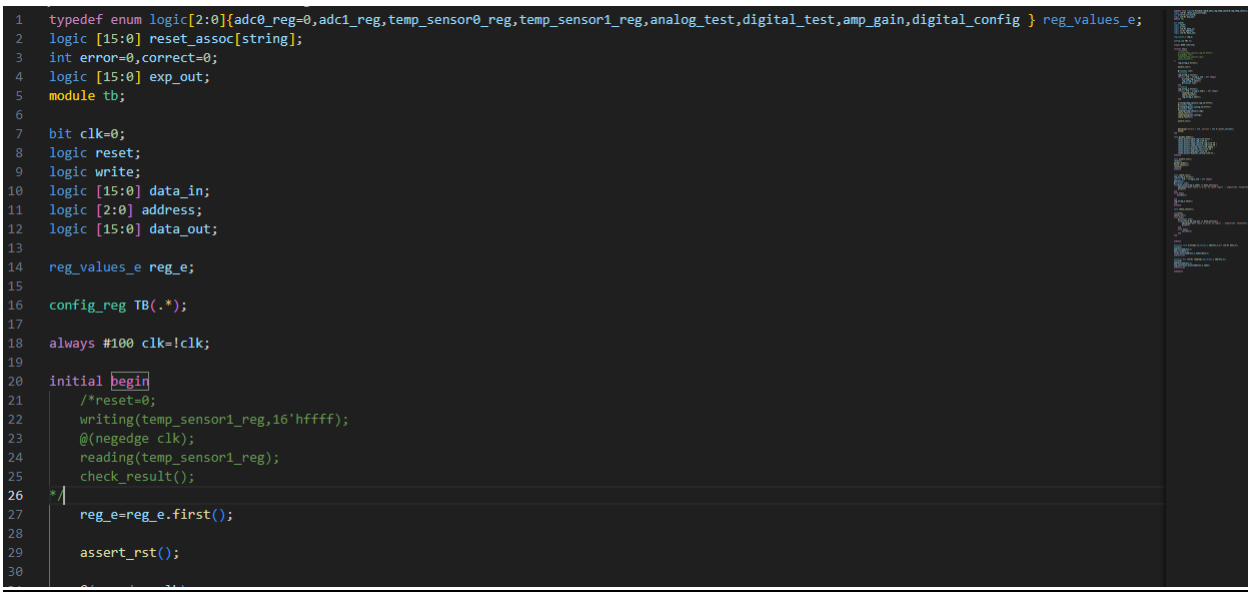
Extra Assigment

Reg_config project

Verification plan

	Label	Description	Stimulus Generation	Functionality Check
2	reg_reset	we assert reset at start of simulation to check the reset values	reset =1 at start of simulation	we check by compare all reg with golden model (associative array)
3	reg_write	here we write values to know if it will be write correctly	write=1 ,data_in with value in for loop and address from enum	-
4	reg_read	we read values of all reg with for loop	write=0, address from enum in for loop	we check by compare all reg with golden model (associative array)
5	reg_reset	we assert reset again to check if value saved in reg not its default value	reset=1	we check by compare all reg with golden model (associative array)

Tb



```
typedef enum logic[2:0]{adc0_reg=0,adc1_reg,temp_sensor0_reg,temp_sensor1_reg,analog_test,digital_test,amp_gain,digital_config } reg_values_e;
logic [15:0] reset_assoc[string];
int error=0,correct=0;
logic [15:0] exp_out;
module tb;

bit clk=0;
logic reset;
logic write;
logic [15:0] data_in;
logic [2:0] address;
logic [15:0] data_out;

reg_values_e reg_e;

config_reg TB(.*);

always #100 clk=!clk;

initial begin
    /*reset=0;
    writing(temp_sensor1_reg,16'hffff);
    @(negedge clk);
    reading(temp_sensor1_reg);
    check_result();
*/
    reg_e=reg_e.first();

    assert_rst();
```

```
    @(negedge clk);
    //writing
    reg_e=reg_e.first();
    for(int x=0 ; x<reg_e.num ; x++ )begin
        writing(reg_e,x+45);
        reg_e=reg_e.next();
        @(negedge clk);
    end
    //reading
    reg_e=reg_e.first();
    for(int x=0 ; x<reg_e.num() ; x++ )begin
        reading(reg_e);
        check_result();
        reg_e=reg_e.next();
    end

    writing(temp_sensor1_reg,16'hffff);
    @(negedge clk);
    writing(digital_config,16'hffff);
    @(negedge clk);
    reading(temp_sensor1_reg);
    check_result();
    reading(digital_config);
    check_result();

    assert_rst();



    $display("errors = %0d ,correct = %0d 0",error,correct);
    $stop;
```

```
end

task golden_model();
    reset_assoc["adc0_reg"]=16'hffff ;
    reset_assoc["adc1_reg"]=16'h0 ;
    reset_assoc["temp_sensor0_reg"]=16'h0 ;
    reset_assoc["temp_sensor1_reg"]=16'h0 ;
    reset_assoc["analog_test"]=16'hABCD ;
    reset_assoc["digital_test"]=16'h0 ;
    reset_assoc["amp_gain"]=16'h0 ;
    reset_assoc["digital_config"]=16'h1 ;
endtask

task assert_rst();
reset=1;
golden_model();
check_result();
reset=0;
endtask


task check_rst();
reg_e=reg_e.first();
for(int i=0 ; i<reg_e.num ; i++ )begin
address=i;
@(negedge clk);
if(reset_assoc[reg_e.name] != data_out)begin
    $display("@%0t there is error on reset reg=%s .. expect=%0h ,found=%0h",$time,reg_e.name,reset_assoc[reg_e.name],data_out);
    error++;
end
else begin
```

```
92  else begin
93      correct++;
94
95    end
96    reg_e=reg_e.next();
97    end
98    endtask
99
100   task check_result();
101
102   if(reset)
103   check_rst();
104 else begin
105       @(negedge clk);
106       if(write==0 && exp_out != data_out)begin
107           $display("@%0t there is error on reg=%s .. expect=%0h ,found=%0h",$time,reg_e.name,exp_out,data_out);
108           error++;
109       end
110       else begin
111           correct++;
112       end
113   end
114
115
116   endtask
117
```

```
118   function void writing(reg_values_e address_t,bit [15:0] data_t);
119   write=1;
120   address=address_t;
121   data_in=data_t;
122   reset_assoc[address_t.name]=data_t;
123   endfunction
124
125   function bit [15:0] reading(reg_values_e address_t);
126   write=0;
127   address=address_t;
128   exp_out=reset_assoc[address_t.name];
129   endfunction
130
131   endmodule
```

Founded buges

Bug 1

a)Design Input for Bug to Appear:

Write input 0x0005 to address 0 (adc0_reg)

b) Expected Behavior:

0x0005 like we write

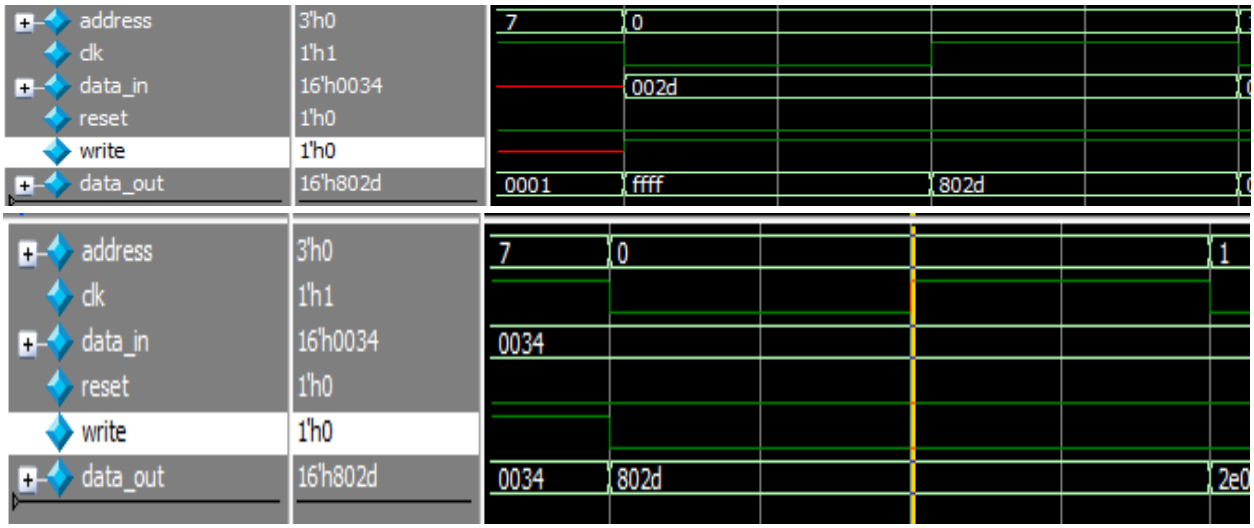
c) Observed Behavior

0x8005 read

d)problem

I think in adc0_reg bit 15 is written always to 0

Simulation



Bug 2

a)Design Input for Bug to Appear:

Write input 0x002a to address 1 (adc1_reg)

b) Expected Behavior:

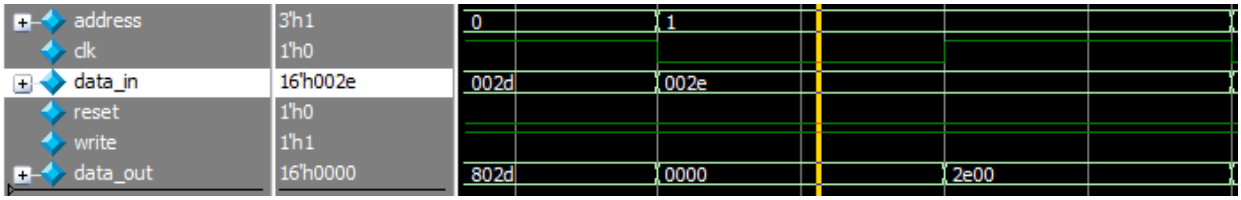
0x002a like we write

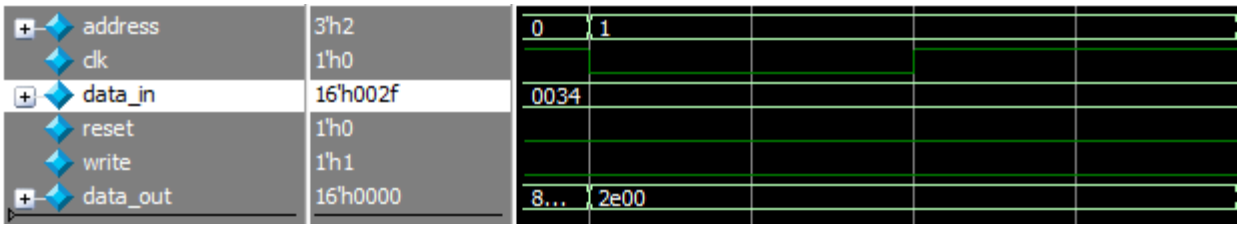
c) Observed Behavior

0x2a00 read

d)problem

I think in adc1_reg it make reverse with 8bit and then packing





Bug 3

a)Design Input for Bug to Appear:

Write input 0x002a to address 2 (temp_sensor0_reg)

b) Expected Behavior:

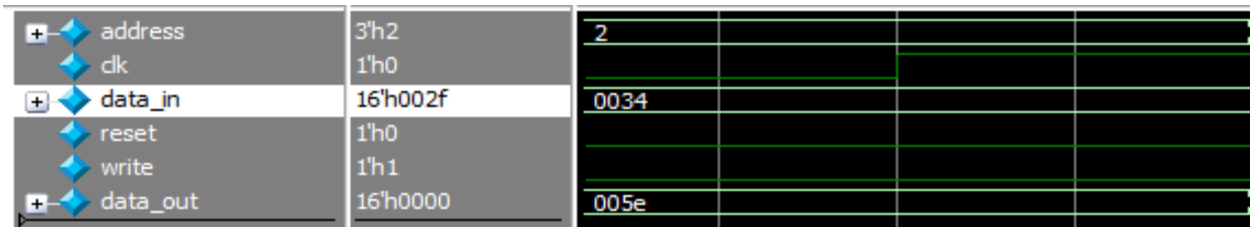
0x002f like we write

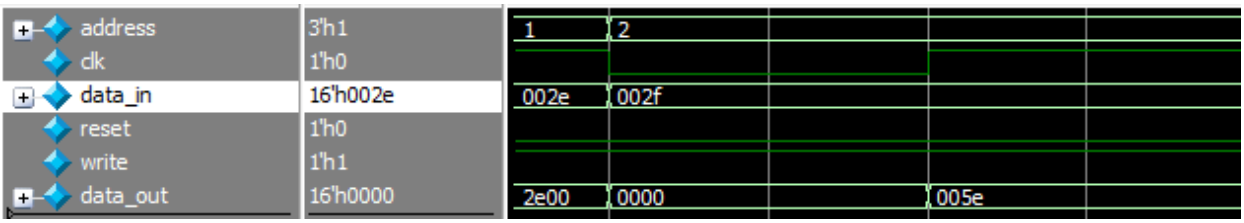
c) Observed Behavior

0x005e read

d)problem

I think in temp_sensor0_reg) it make shift left for data_in





Bug 4

a)Design Input for Bug to Appear:

assert reset

b) Expected Behavior:

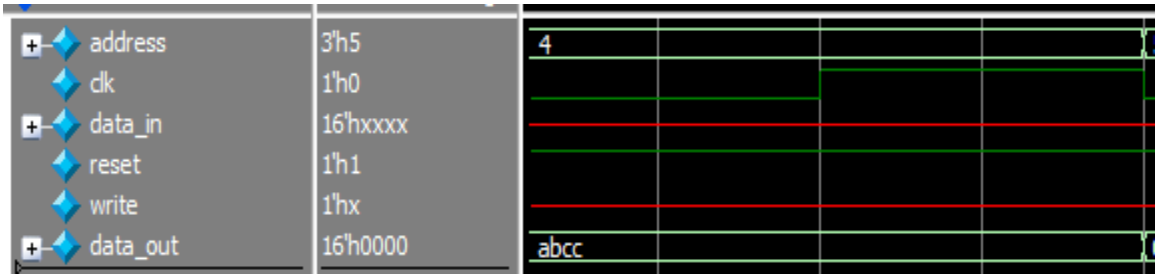
analog_test reg in reset =ABCD

c) Observed Behavior

analog_test reg in reset =ABCC

d)problem

wrong initialize



Bug 5

a)Design Input for Bug to Appear:

Write input 0x0032 to address 5 (digital_test)

b) Expected Behavior:

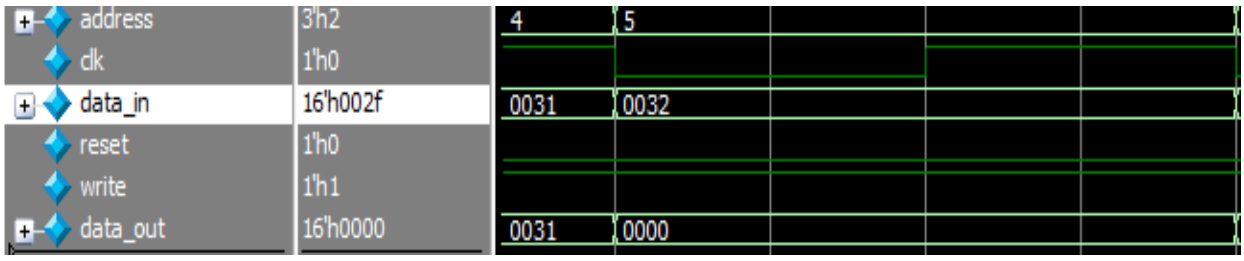
0x0032 like we write

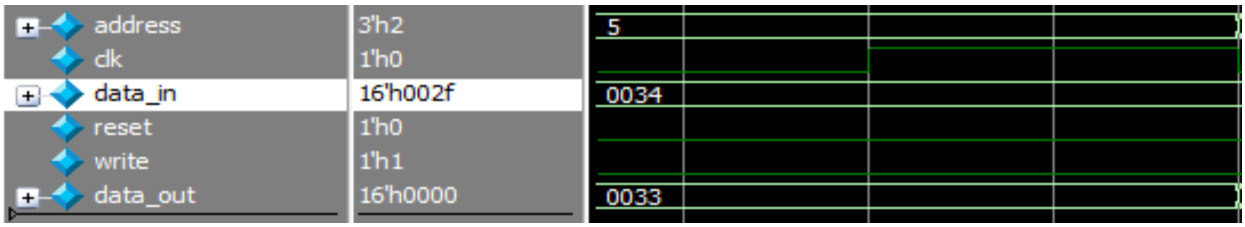
c) Observed Behavior

0x0033read

d)problem

I think its take wrong addres 6 instead of 5





Bug 6

a)Design Input for Bug to Appear:

Write input 0x0033 to address 6 (amp_gain)

b) Expected Behavior:

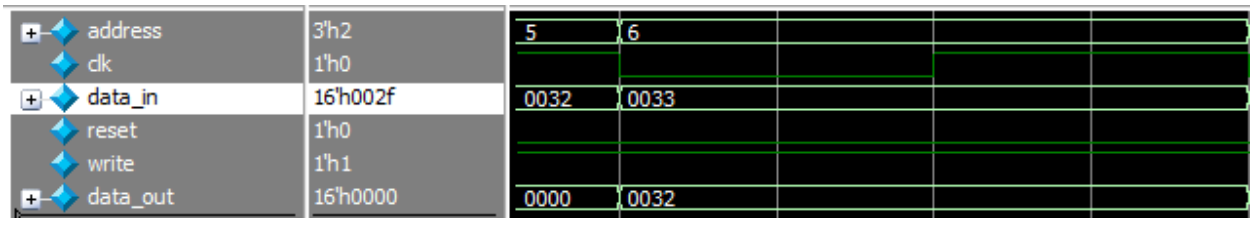
0x0033 like we write

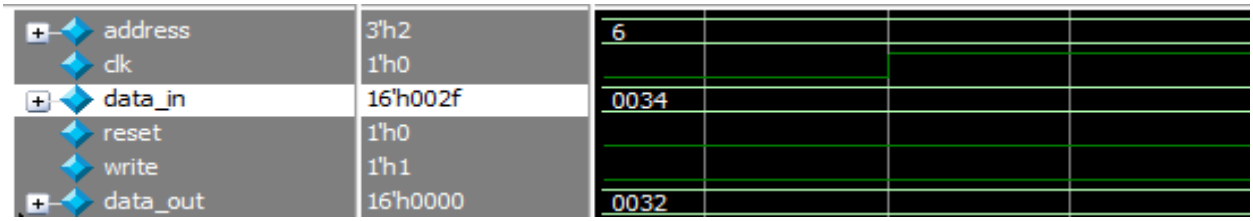
c) Observed Behavior

0x0032 read

d)problem

I think its take wrong addres 5 instead of 6





Bug 7

a)Design Input for Bug to Appear:

Write input 0xffff to address 6 (digital_config)

b) Expected Behavior:

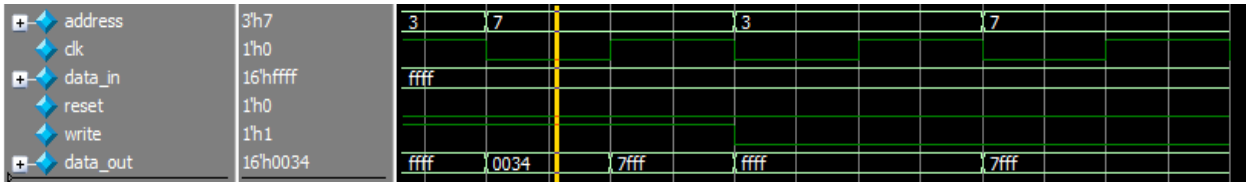
0xffff like we write

c) Observed Behavior

0x7fff read

d)problem

I think this reg is only 15 bits



Bug 8

a)Design Input for Bug to Appear:

Write input 0xffff to address 6 (temp_sensor1_reg) then we reset

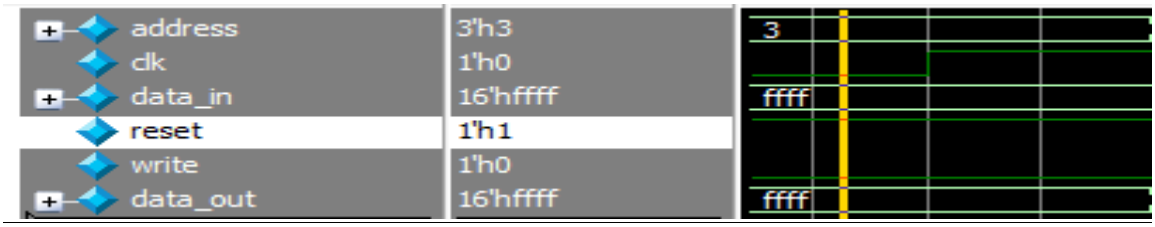
b) Expected Behavior:

0x0000 as reset value

c) Observed Behavior 0xffff read

d)problem

I think this reg is bit not logic so if we assert reset at beginning it show like its implementation is right but only it s default value



Arbiter

```
property prop_1;
    @(posedge clk)($rose(request) |-> ##[2:5] grant);
endproperty

property prop_2;
@(posedge clk)( $rose(grant) |-> (!frame && !irdy) );
endproperty

property prop_3;
@(posedge clk)  ((frame && irdy) |=> !grant);
endproperty
```

FSM

```
property fsm1;// if we dont have a clock
@(cs) $onehot(cs);
endproperty



property fsm2;
@(posedge clk) (cs==IDLE && get_data)|=> (cs==GEN_BLK_ADDR) ##[1:64] (cs==WAITO);
endproperty
```