به نام خدا

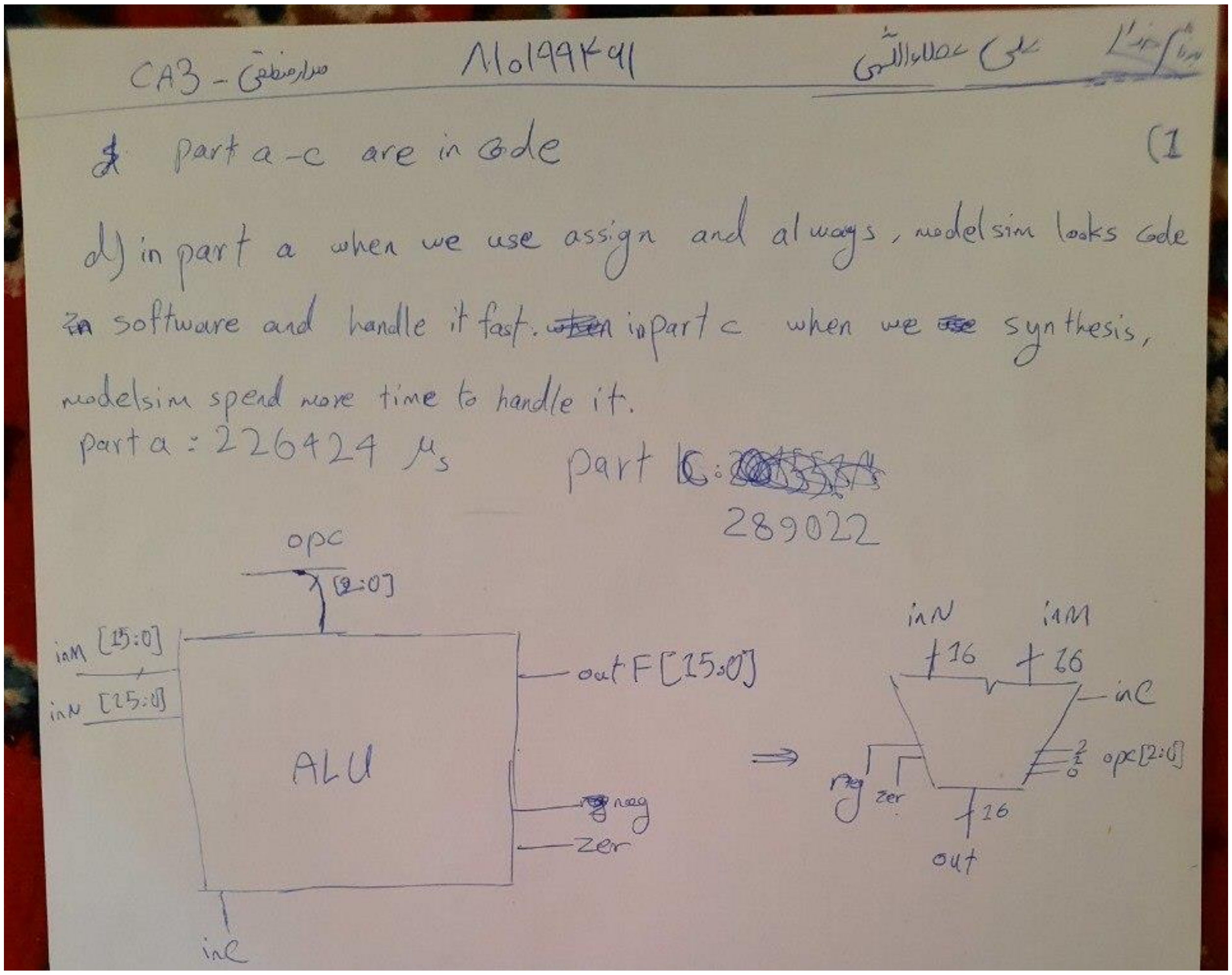
علی عطاالللهی

810199461

:1

For part d we use

```
set runtime [time {vsim -do "run -all" my_toplevel}]
```



به نام خدا   علی عطاءاللهی   ۸۱۰۱۹۹۴۶۱   CA3 - مدار منطقی

(1

part a-c are in code

d) in part a when we use assign and always, modelsim looks code ~~in~~ software and handle it fast. ~~when~~ inpart c when we ~~use~~ synthesis, modelsim spend more time to handle it.

part a : 226424 μ_s

part ~~b~~ c: ~~[illegible]~~ 289022

opc
[2:0]
inM [15:0]
inN [15:0]
ALU
out F[15:0]
neg
zer
inC
⇒
inN
inM
16
16
inC
opc[2:0]
neg
zer
16
out

`timescale 1ns/1ns

```
module ALU1 (input signed [15:0] inM,inN,input [2:0] opc,input inC,output zer,neg,output logic
signed [15:0] outF);
        assign zer=~(|outF);
        assign neg=outF[15];
        always @(inM,inN,opc,inC) begin
                outF=16'd0;
                case(opc)
                        3'd0:outF=inM+inN+inC;
                        3'd1:outF=inM+(inN>>>1);
                        3'd2:outF=inM+1;
                        3'd3:outF=inM+(inM>>>1);
                        3'd4:outF=inM&inN;
                        3'd5:outF=inM|inN;
                        3'd6:outF=~inM;
                        3'd7:outF=16'd0;
                        default:outF=16'd0;
                endcase
        end
endmodule
```

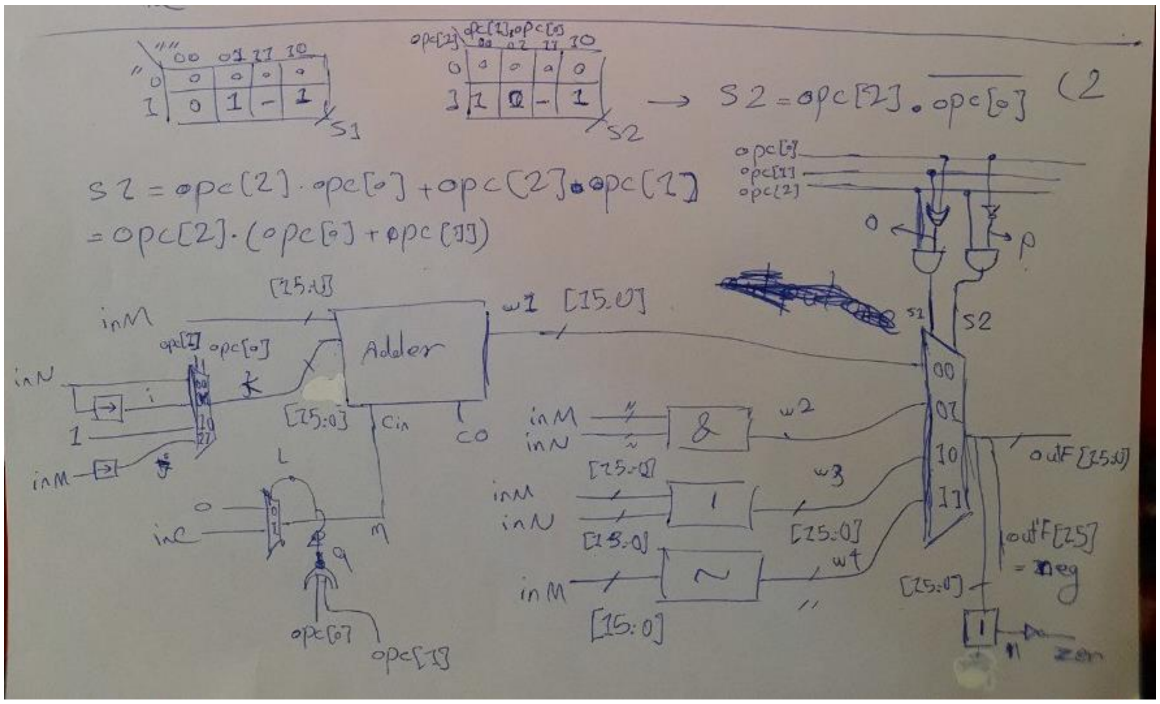
testbench:

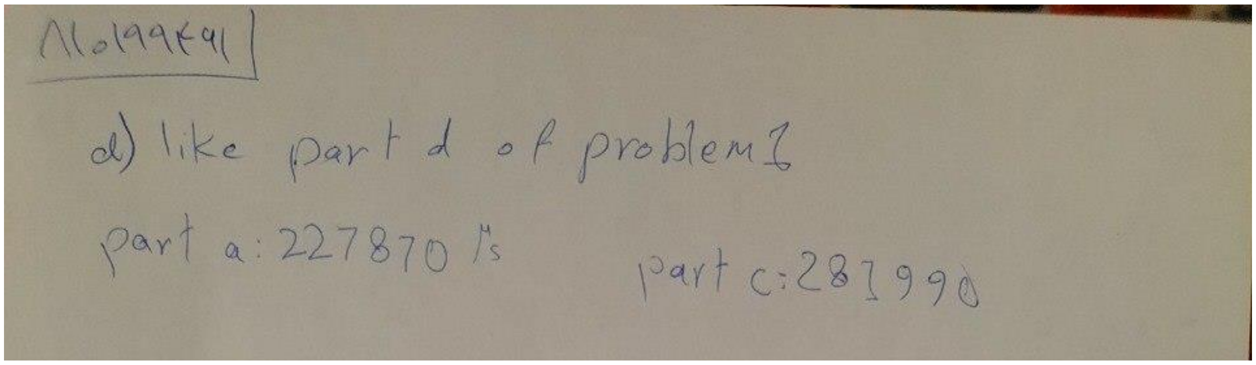
```
`timescale 1ns/1ns
module CA3_E1_TB ();
```

```
logic [15:0] inMM=3'd5,inNN=3'd4;

logic [2:0] opcc;

logic inCC=1'b0;

wire zerr1,negg1,zerr2,negg2;

logic [15:0] outFF1,outFF2;

ALU1 alu1(inMM,inNN,opcc,inCC,zerr1,negg1,outFF1);

ALU2 alu2(inMM,inNN,opcc,inCC,zerr2,negg2,outFF2);

initial begin

opcc=3'd0;

repeat (10) begin

        #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;

end

opcc=3'd1;

repeat (10) begin

        #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;

end

opcc=3'd2;

repeat (10) begin

        #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;

end

opcc=3'd3;

repeat (10) begin
```

```
        #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;
    end
    opcc=3'd4;
    repeat (10) begin
        #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;
    end
    opcc=3'd5;
    repeat (10) begin
        #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;
    end
    opcc=3'd6;
    repeat (10) begin
        #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;
    end
    #1000 $stop;
    end
endmodule
```

2:





810199491

d) like part d of problem 1

part a: 227870 ps

part c: 281990

```
`timescale 1ns/1ns

module ALU_STR1 (input signed [15:0] inM,inN,input [2:0] opc,input inC,output zer,neg,output
signed [15:0] outF);

        wire [15:0] w1,w2,w3,w4,i,j,k;

        wire l,m,n,co,s1,s2,o,p;

        assign neg=outF[15];

        logic [15:0] one=16'b1;

        logic zero=1'b0;
```

```
	or or1(o,opc[1],opc[0]);

	not not1(p,opc[0]);

	and and1(s1,opc[2],o);

	and and2(s2,opc[2],p);

	mux16bit mux1(w1,w2,w3,w4,{s1,s2},outF);

	allor a1(outF,n);

	not not2(zer,n);

	adder16bits adder(inM,k,m,co,w1);

	and16 a2(inM,inN,w2);

	or16 a3(inM,inN,w3);

	allinv a4(inM,w4);

	mux16bit mux2(inN,i,one,j,{opc[1],opc[0]},k);

	shifter sh1(inN,i);

	shifter sh2(inM,j);

	mux2to1 mux3(zero,inC,l,m);

	or or2(q,opc[0],opc[1]);

	not not3(l,q);

endmodule

module shifter (input signed [15:0] in,output signed [15:0] out);

	assign out=in>>>1;

endmodule

module adder16bits (input signed [15:0] in1,in2,input cin,output co,output signed [15:0] out);
```

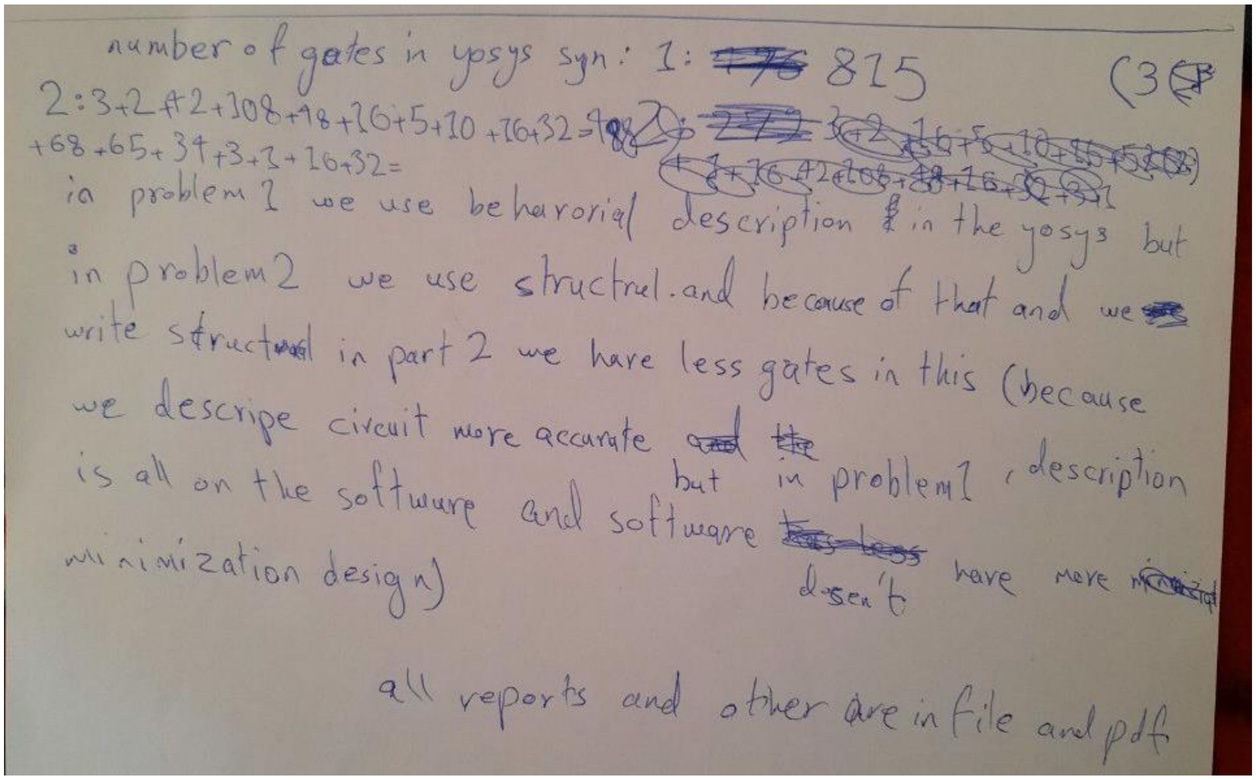
```
        assign {co,out}=in1+in2+cin;

endmodule

module mux16bit (input [15:0] a,b,c,d,input [1:0] select,output [15:0] outF);

        assign outF =   (select==2'd0) ? a:

                        (select==2'd1) ? b:

                        (select==2'd2) ? c:

                        (select==2'd3) ? d:16'bx;

endmodule

module mux2to1 (input a,b,select,output outF);

        assign outF =   (select==1'b0) ? a:

                        (select==1'b1) ? b:1'bx;

endmodule

module and16 (input [15:0] a,b,output [15:0] outF);

        assign outF = a&b;

endmodule

module or16 (input [15:0] a,b,output [15:0] outF);

        assign outF = a|b;

endmodule

module allor (input [15:0] a,output outF);

        assign outF = |a;

endmodule

module allinv (input [15:0] a,output [15:0] outF);
```

```
	assign outF=~a;
endmodule



testbench
`timescale 1ns/1ns
module CA3_E2_TB ();
	logic [15:0] inMM=3'd5,inNN=3'd4;
	logic [2:0] opcc;
	logic inCC=1'b0;
	wire zerr1,negg1,zerr2,negg2;
	logic [15:0] outFF1,outFF2;
	ALU_STR1 alu1(inMM,inNN,opcc,inCC,zerr1,negg1,outFF1);
	ALU_STR2 alu2(inMM,inNN,opcc,inCC,zerr2,negg2,outFF2);
	initial begin
	opcc=3'd0;
	repeat (10) begin
		#40 inMM=$random; #40 inNN=$random; #20 inCC=$random;
	end
	opcc=3'd1;
	repeat (10) begin
		#40 inMM=$random; #40 inNN=$random; #20 inCC=$random;
```

```
end

opcc=3'd2;

repeat (10) begin

            #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;

end

opcc=3'd3;

repeat (10) begin

            #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;

end

opcc=3'd4;

repeat (10) begin

            #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;

end

opcc=3'd5;

repeat (10) begin

            #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;

end

opcc=3'd6;

repeat (10) begin

            #40 inMM=$random; #40 inNN=$random; #20 inCC=$random;

end

#1000 $stop;
```

```
        end

endmodule
```

3:

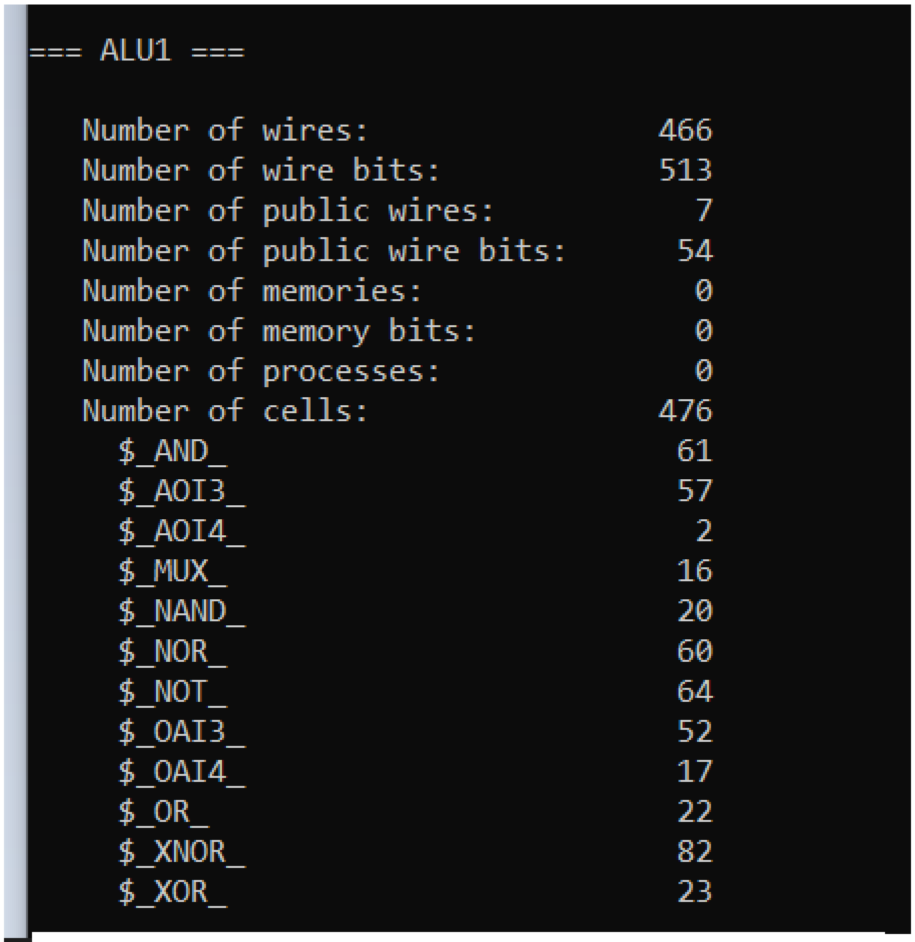


number of gates in yosys syn: 1: ~~176~~ 815 (3 [illegible]

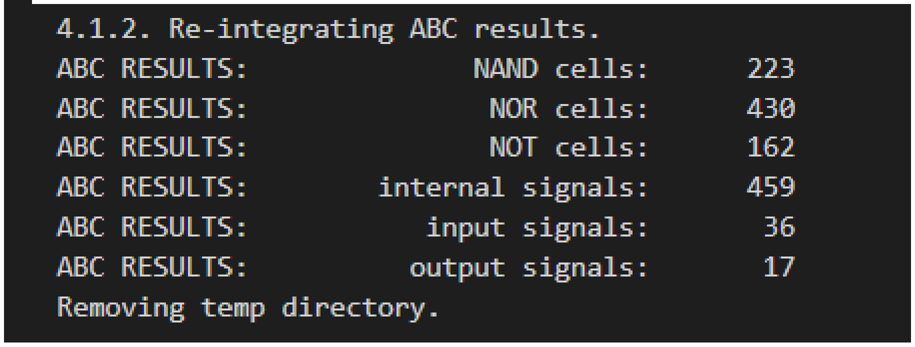
2: 3+2+2+108+98+16+5+10+16+32=[illegible] ~~2~~ : ~~277~~ ~~3+2+16+5+10+16+5+[illegible]~~
+68+65+34+3+3+16+32= ~~+3+16+42+108+98+16+32+3~~

in problem 1 we use beharorial description ~~&~~ in the yosys but in problem2 we use structrel. and because of that and we ~~[illegible]~~ write structral in part 2 we have less gates in this (because we descripe circuit more accurate ~~and~~ ~~the~~ but in problem1, description is all on the software and software ~~less less~~ dosen't have more ~~[illegible]~~ minimization design)

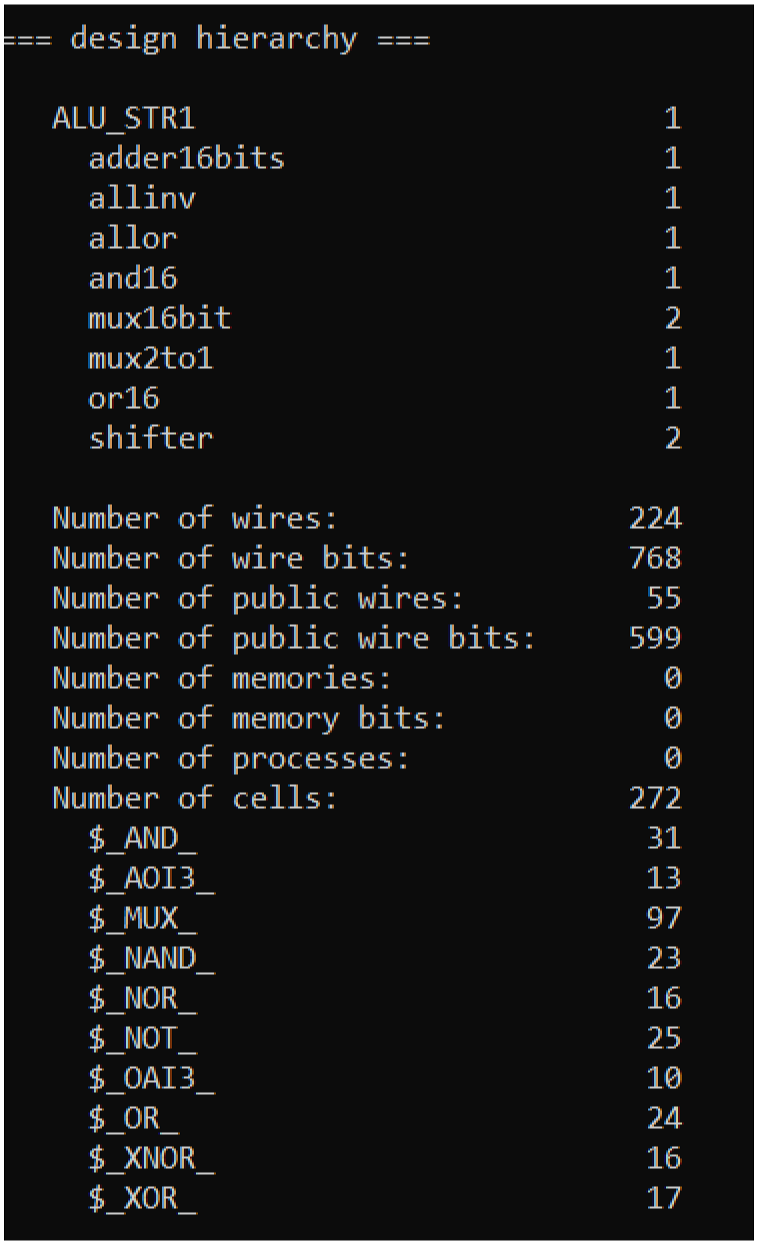
all reports and other are in file and pdf



```
=== ALU1 ===

   Number of wires:                466
   Number of wire bits:            513
   Number of public wires:           7
   Number of public wire bits:      54
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                476
     $_AND_                         61
     $_AOI3_                        57
     $_AOI4_                         2
     $_MUX_                         16
     $_NAND_                        20
     $_NOR_                         60
     $_NOT_                         64
     $_OAI3_                        52
     $_OAI4_                        17
     $_OR_                          22
     $_XNOR_                        82
     $_XOR_                         23
```



```
4.1.2. Re-integrating ABC results.
ABC RESULTS:              NAND cells:       223
ABC RESULTS:               NOR cells:       430
ABC RESULTS:               NOT cells:       162
ABC RESULTS:        internal signals:       459
ABC RESULTS:           input signals:        36
ABC RESULTS:          output signals:        17
Removing temp directory.
```



```
=== design hierarchy ===

   ALU_STR1                          1
     adder16bits                     1
     allinv                          1
     allor                           1
     and16                           1
     mux16bit                        2
     mux2to1                         1
     or16                            1
     shifter                         2

   Number of wires:                224
   Number of wire bits:            768
   Number of public wires:          55
   Number of public wire bits:     599
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                272
     $_AND_                         31
     $_AOI3_                        13
     $_MUX_                         97
     $_NAND_                        23
     $_NOR_                         16
     $_NOT_                         25
     $_OAI3_                        10
     $_OR_                          24
     $_XNOR_                        16
     $_XOR_                         17
```