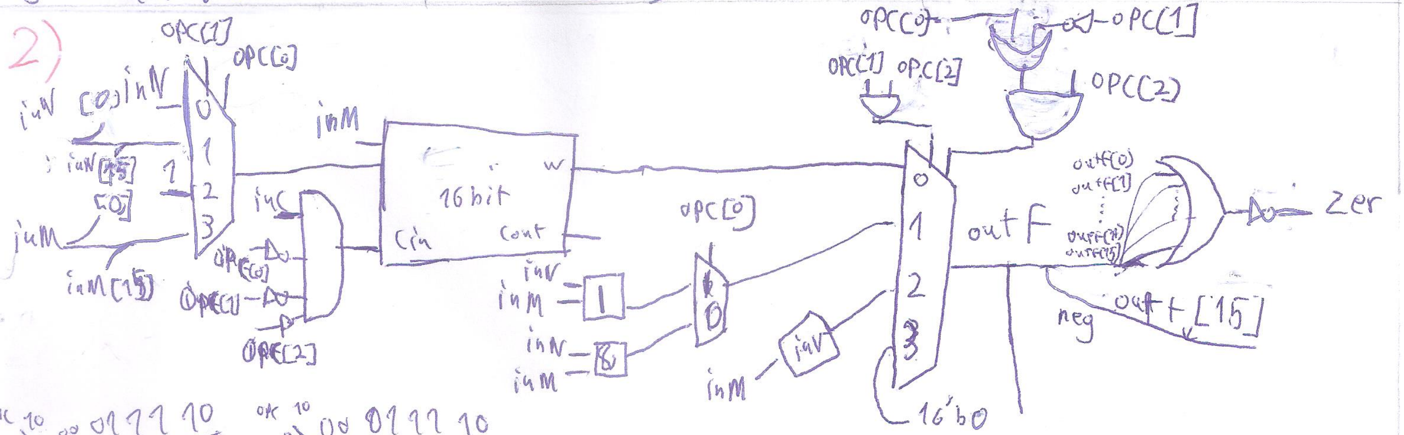
Arash Shahin
8101992

Digital Logic Design
CA3

1400/9/10
ECE367

2)



opc 2 \ 10	00	01	11	10
0	0	0	0	0
1	1	1	1	0

f0

opc 2 \ 10	00	01	11	10
0	0	0	0	0
1	0	0	1	1

f1

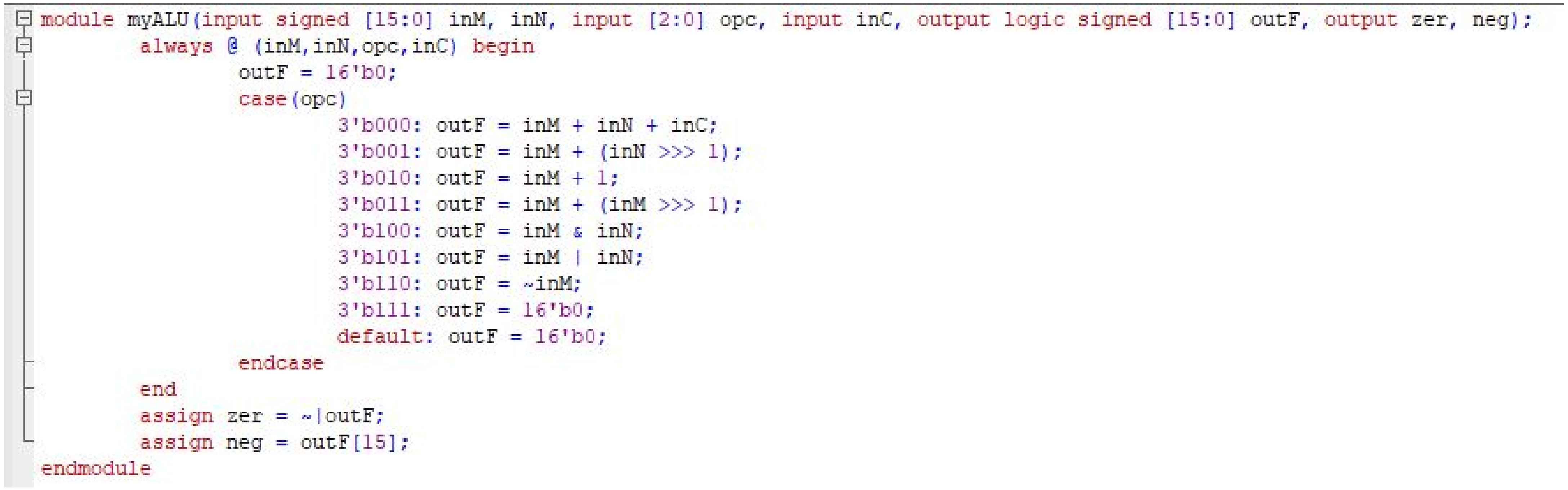
For the last multiplezer

3) Ourresults will bethe same. However, in the first part, since we did not use a hardware based design but a more software based one, Yosys used more gates. While the hardware based one used less gates. So we used less gates in the hardware based design and so we'll have a shorter delay and a higher simulation speed.

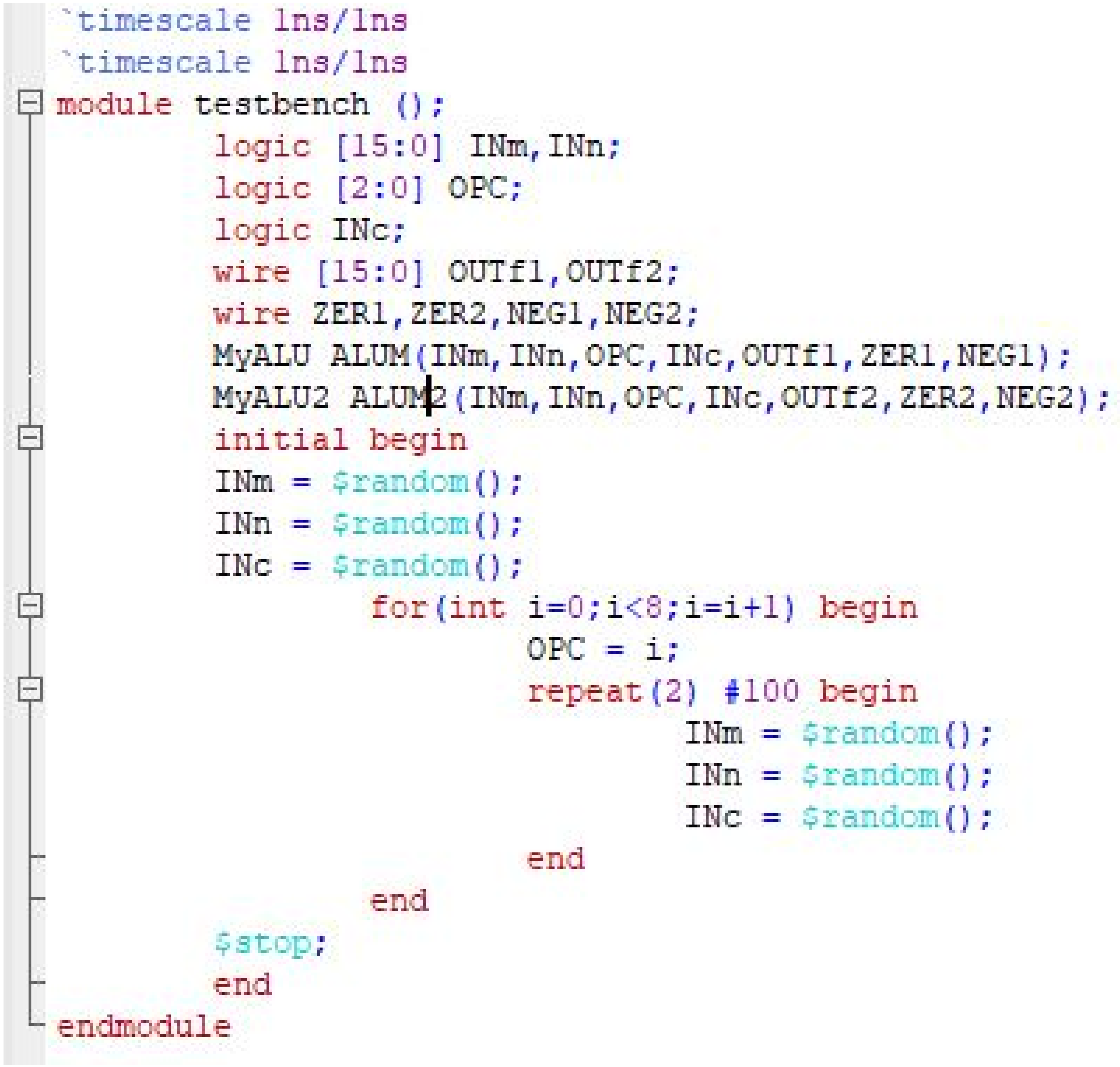
1,2)d) simulation speed is allmost equal altlough yosys is slighty slower because Yosys will use the gates it's synthsised while our code will use the computer's ~~[illegible]~~ more optimized hardware

part1 yosys gates: 223 NAND / 430 NOR / 162 NOT
Total: 815 gates

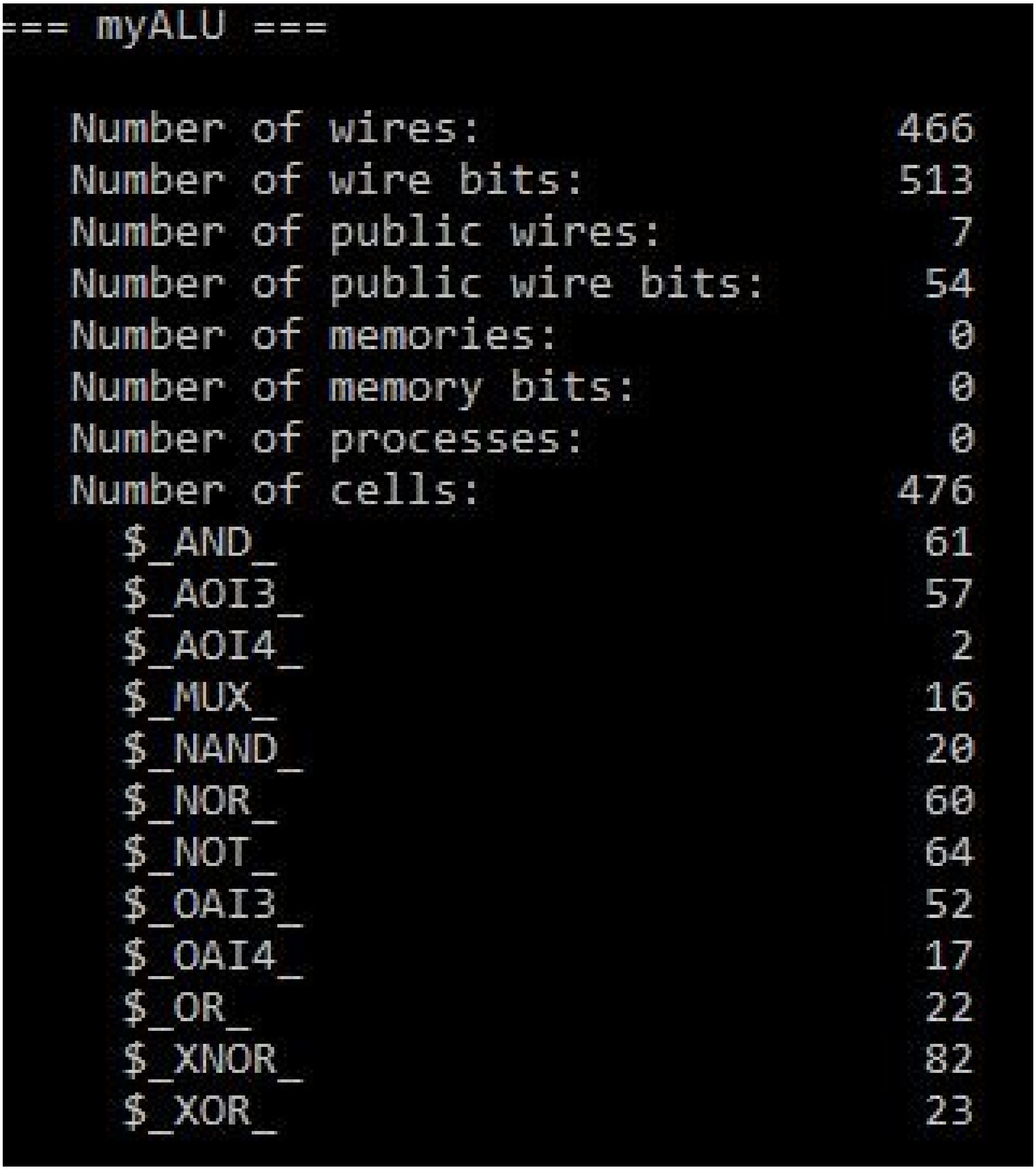
Part2 yosys gates: 142 NAND / 238 NOR / 138 NOT
Total: 518 gates



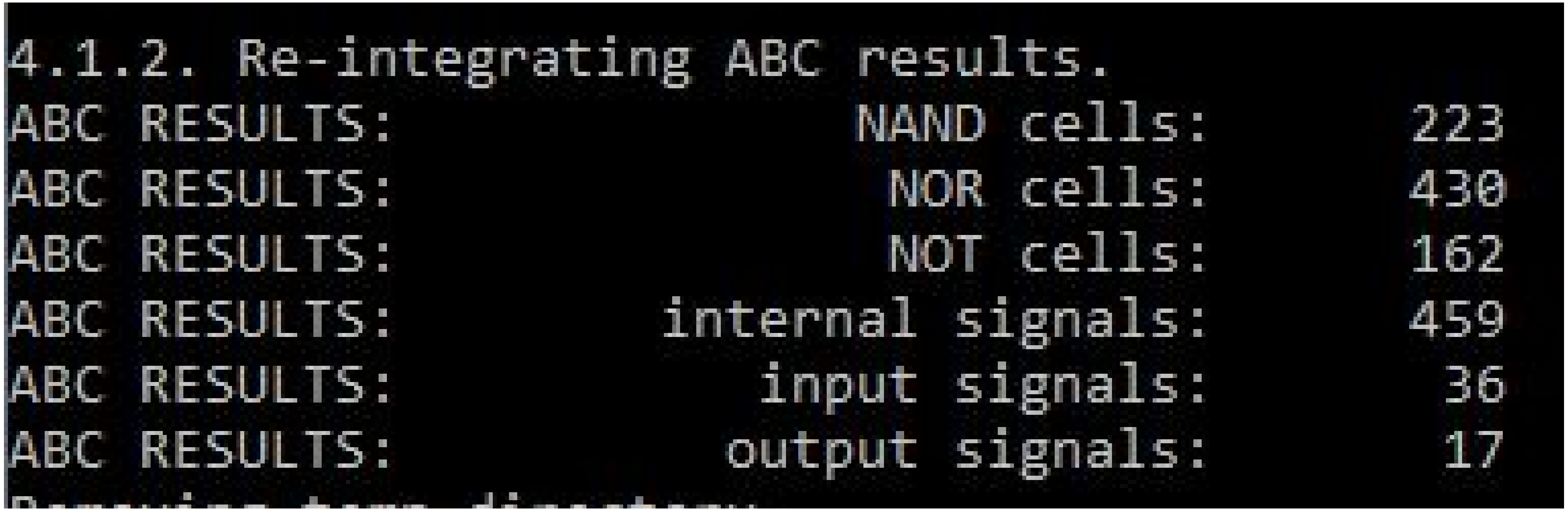
```
module myALU(input signed [15:0] inM, inN, input [2:0] opc, input inC, output logic signed [15:0] outF, output zer, neg);
        always @ (inM,inN,opc,inC) begin
                outF = 16'b0;
                case(opc)
                        3'b000: outF = inM + inN + inC;
                        3'b001: outF = inM + (inN >>> 1);
                        3'b010: outF = inM + 1;
                        3'b011: outF = inM + (inM >>> 1);
                        3'b100: outF = inM & inN;
                        3'b101: outF = inM | inN;
                        3'b110: outF = ~inM;
                        3'b111: outF = 16'b0;
                        default: outF = 16'b0;
                endcase
        end
        assign zer = ~|outF;
        assign neg = outF[15];
endmodule
```



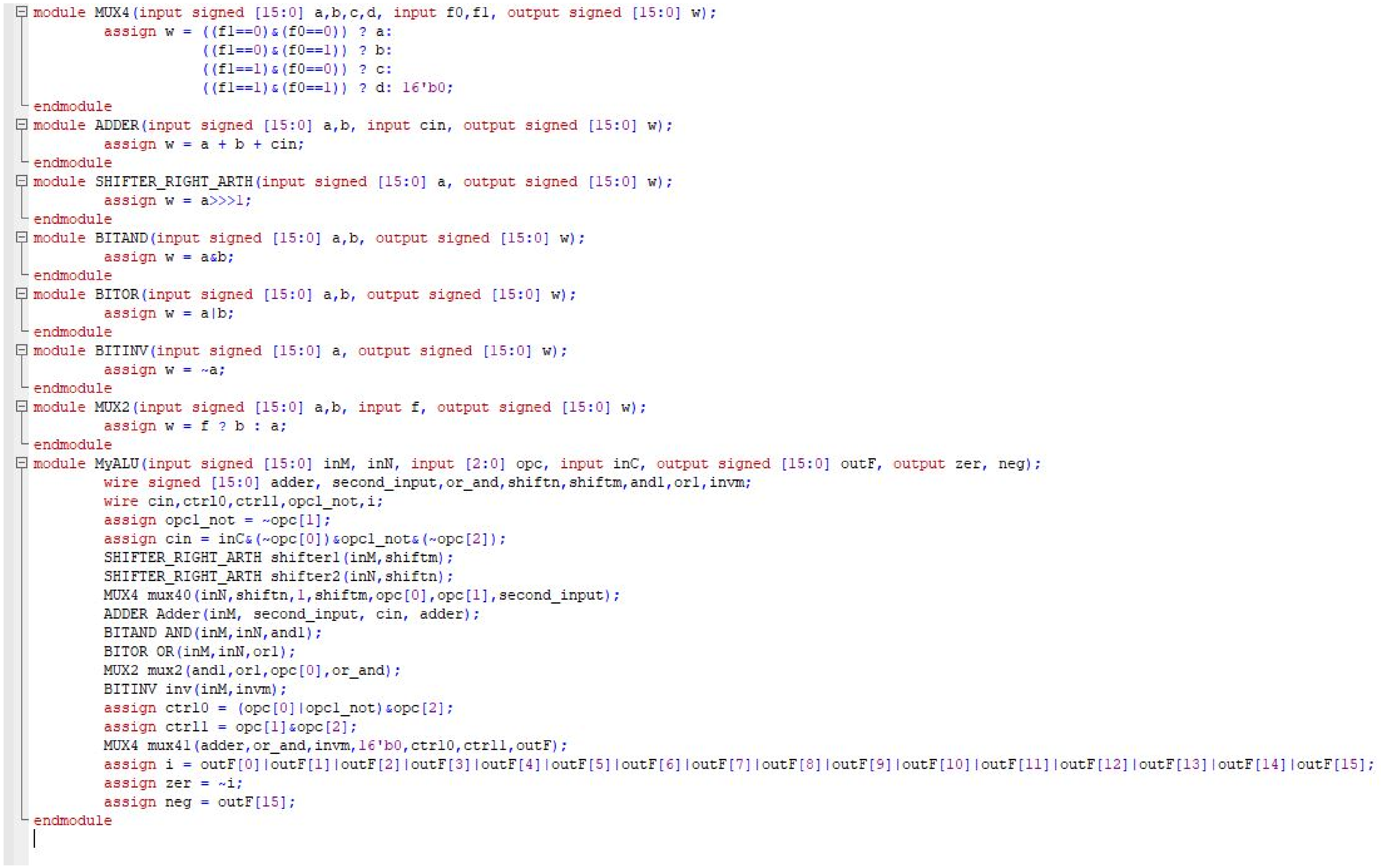
```
`timescale 1ns/1ns
`timescale 1ns/1ns
module testbench ();
        logic [15:0] INm,INn;
        logic [2:0] OPC;
        logic INc;
        wire [15:0] OUTf1,OUTf2;
        wire ZER1,ZER2,NEG1,NEG2;
        MyALU ALUM(INm,INn,OPC,INc,OUTf1,ZER1,NEG1);
        MyALU2 ALUM2(INm,INn,OPC,INc,OUTf2,ZER2,NEG2);
        initial begin
        INm = $random();
        INn = $random();
        INc = $random();
                for(int i=0;i<8;i=i+1) begin
                        OPC = i;
                        repeat(2) #100 begin
                                INm = $random();
                                INn = $random();
                                INc = $random();
                        end
                end
        $stop;
        end
endmodule
```

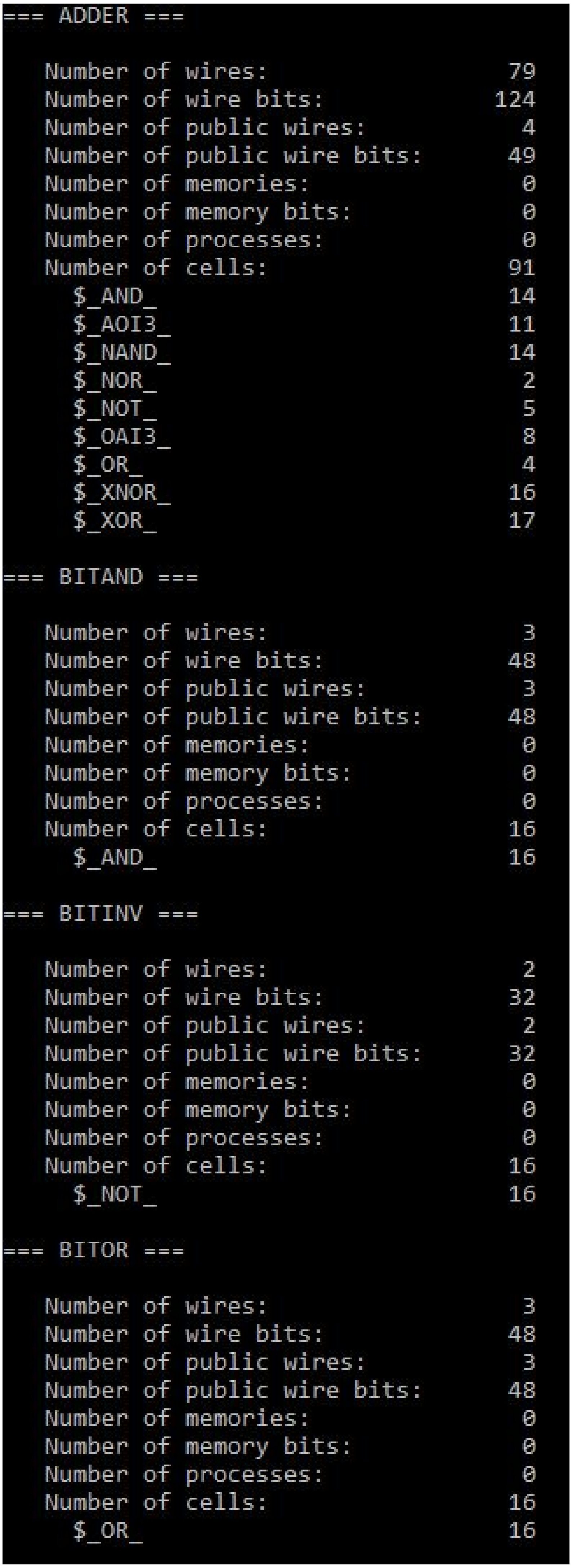


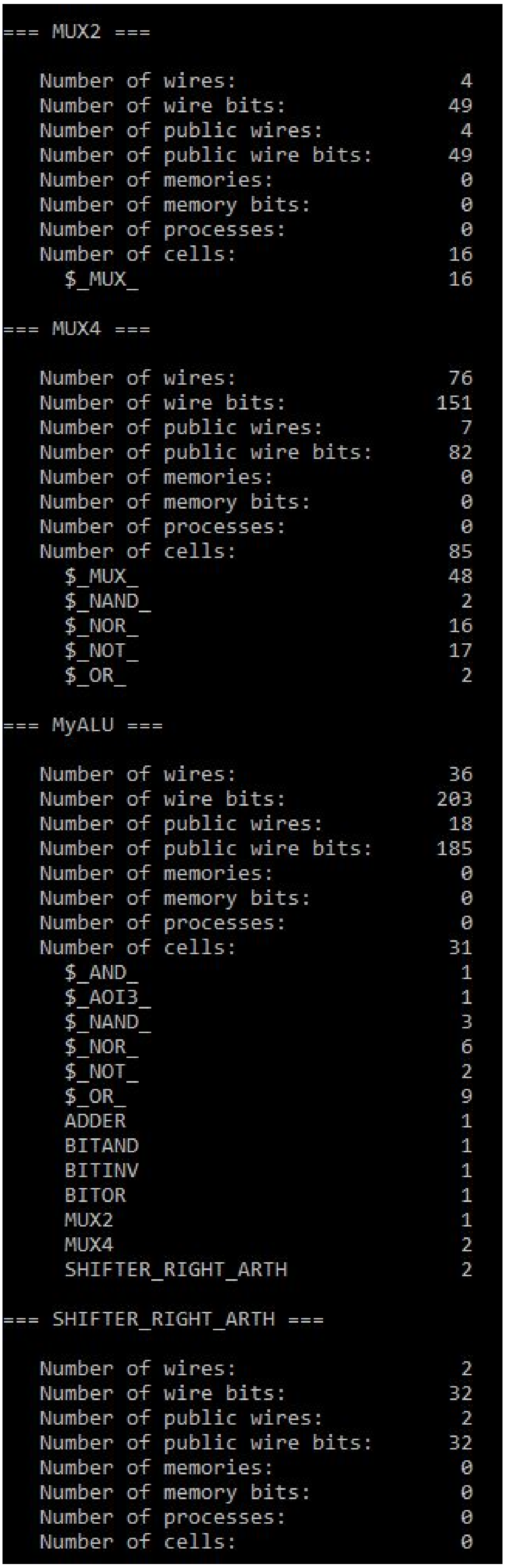
```
=== myALU ===

   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
```



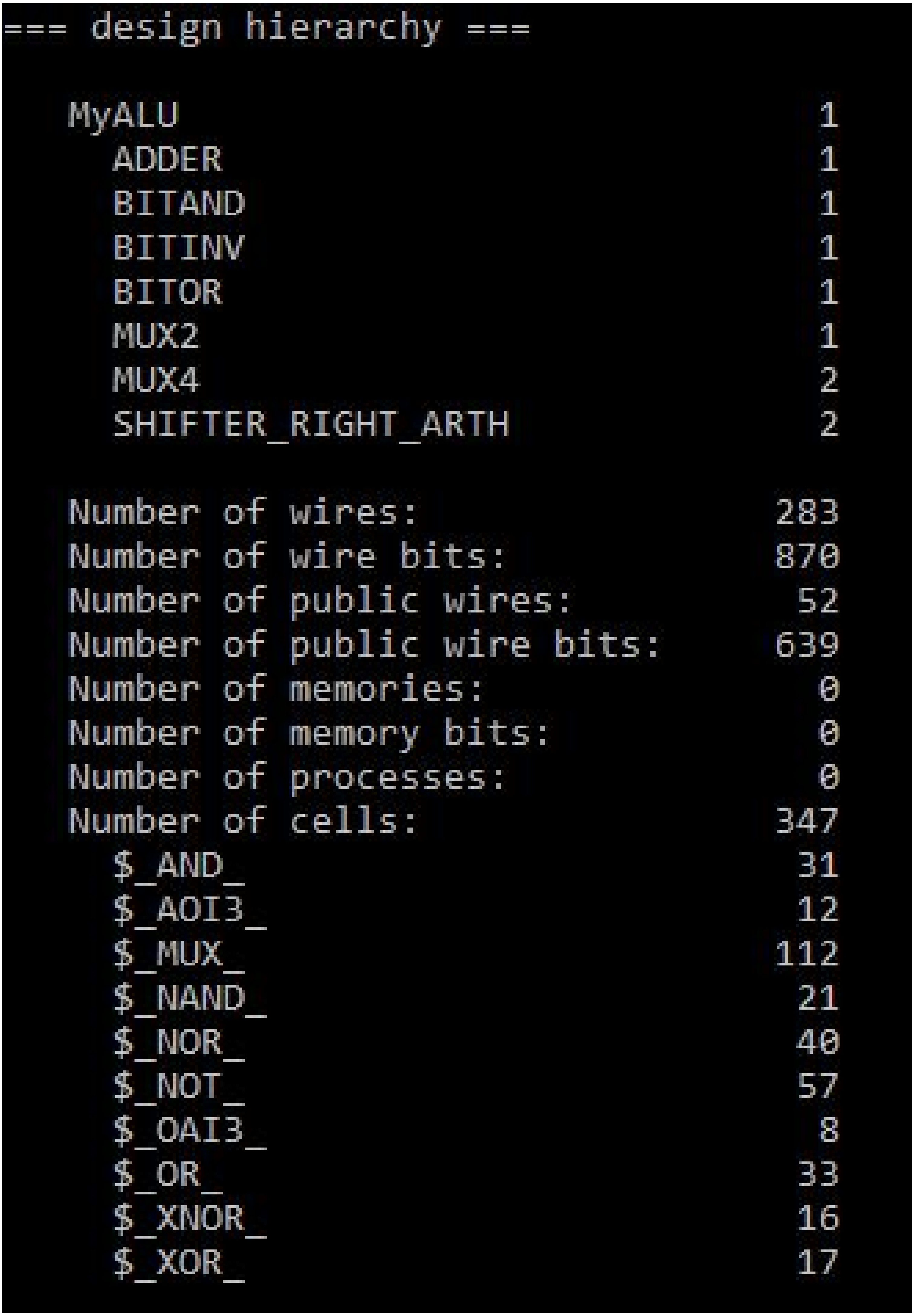
```
4.1.2. Re-integrating ABC results.
ABC RESULTS:               NAND cells:       223
ABC RESULTS:                NOR cells:       430
ABC RESULTS:                NOT cells:       162
ABC RESULTS:         internal signals:       459
ABC RESULTS:            input signals:        36
ABC RESULTS:    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

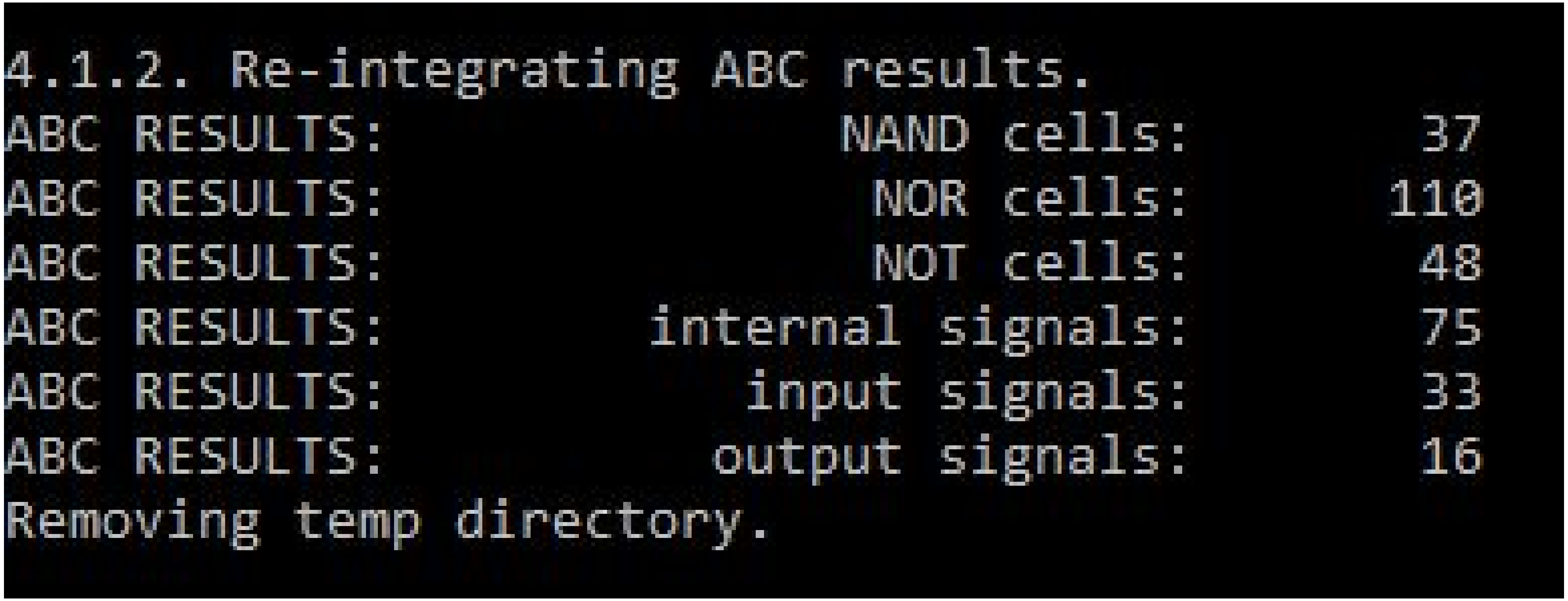

```verilog
module MUX4(input signed [15:0] a,b,c,d, input f0,f1, output signed [15:0] w);
        assign w = ((f1==0)&(f0==0)) ? a:
                   ((f1==0)&(f0==1)) ? b:
                   ((f1==1)&(f0==0)) ? c:
                   ((f1==1)&(f0==1)) ? d: 16'b0;
endmodule
module ADDER(input signed [15:0] a,b, input cin, output signed [15:0] w);
        assign w = a + b + cin;
endmodule
module SHIFTER_RIGHT_ARTH(input signed [15:0] a, output signed [15:0] w);
        assign w = a>>>1;
endmodule
module BITAND(input signed [15:0] a,b, output signed [15:0] w);
        assign w = a&b;
endmodule
module BITOR(input signed [15:0] a,b, output signed [15:0] w);
        assign w = a|b;
endmodule
module BITINV(input signed [15:0] a, output signed [15:0] w);
        assign w = ~a;
endmodule
module MUX2(input signed [15:0] a,b, input f, output signed [15:0] w);
        assign w = f ? b : a;
endmodule
module MyALU(input signed [15:0] inM, inN, input [2:0] opc, input inC, output signed [15:0] outF, output zer, neg);
        wire signed [15:0] adder, second_input,or_and,shiftn,shiftm,and1,or1,invm;
        wire cin,ctrl0,ctrl1,opc1_not,i;
        assign opc1_not = ~opc[1];
        assign cin = inC&(~opc[0])&opc1_not&(~opc[2]);
        SHIFTER_RIGHT_ARTH shifter1(inM,shiftm);
        SHIFTER_RIGHT_ARTH shifter2(inN,shiftn);
        MUX4 mux40(inN,shiftn,1,shiftm,opc[0],opc[1],second_input);
        ADDER Adder(inM, second_input, cin, adder);
        BITAND AND(inM,inN,and1);
        BITOR OR(inM,inN,or1);
        MUX2 mux2(and1,or1,opc[0],or_and);
        BITINV inv(inM,invm);
        assign ctrl0 = (opc[0]|opc1_not)&opc[2];
        assign ctrl1 = opc[1]&opc[2];
        MUX4 mux41(adder,or_and,invm,16'b0,ctrl0,ctrl1,outF);
        assign i = outF[0]|outF[1]|outF[2]|outF[3]|outF[4]|outF[5]|outF[6]|outF[7]|outF[8]|outF[9]|outF[10]|outF[11]|outF[12]|outF[13]|outF[14]|outF[15];
        assign zer = ~i;
        assign neg = outF[15];
endmodule
```


```
=== ADDER ===

   Number of wires:                 79
   Number of wire bits:            124
   Number of public wires:           4
   Number of public wire bits:      49
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 91
     $_AND_                         14
     $_AOI3_                        11
     $_NAND_                        14
     $_NOR_                          2
     $_NOT_                          5
     $_OAI3_                         8
     $_OR_                           4
     $_XNOR_                        16
     $_XOR_                         17

=== BITAND ===

   Number of wires:                  3
   Number of wire bits:             48
   Number of public wires:           3
   Number of public wire bits:      48
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 16
     $_AND_                         16

=== BITINV ===

   Number of wires:                  2
   Number of wire bits:             32
   Number of public wires:           2
   Number of public wire bits:      32
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 16
     $_NOT_                         16

=== BITOR ===

   Number of wires:                  3
   Number of wire bits:             48
   Number of public wires:           3
   Number of public wire bits:      48
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 16
     $_OR_                          16
```

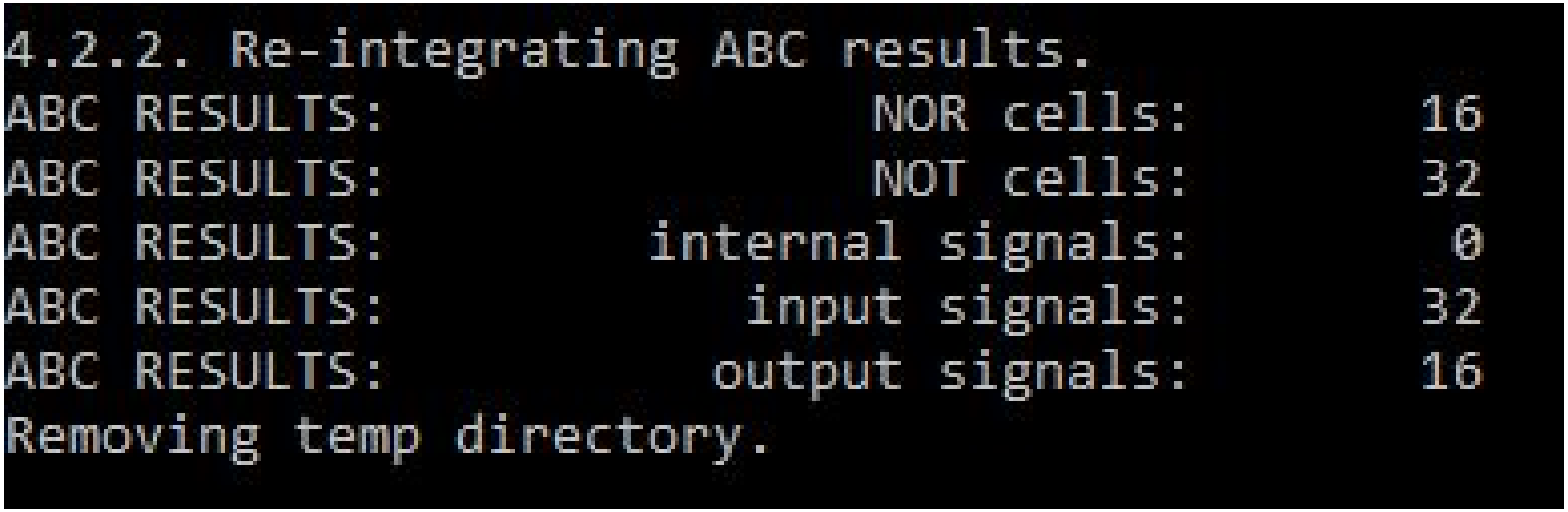


```
=== MUX2 ===

   Number of wires:                  4
   Number of wire bits:             49
   Number of public wires:           4
   Number of public wire bits:      49
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 16
     $_MUX_                         16

=== MUX4 ===

   Number of wires:                 76
   Number of wire bits:            151
   Number of public wires:           7
   Number of public wire bits:      82
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 85
     $_MUX_                         48
     $_NAND_                         2
     $_NOR_                         16
     $_NOT_                         17
     $_OR_                           2

=== MyALU ===

   Number of wires:                 36
   Number of wire bits:            203
   Number of public wires:          18
   Number of public wire bits:     185
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 31
     $_AND_                          1
     $_AOI3_                         1
     $_NAND_                         3
     $_NOR_                          6
     $_NOT_                          2
     $_OR_                           9
     ADDER                           1
     BITAND                          1
     BITINV                          1
     BITOR                           1
     MUX2                            1
     MUX4                            2
     SHIFTER_RIGHT_ARTH              2

=== SHIFTER_RIGHT_ARTH ===

   Number of wires:                  2
   Number of wire bits:             32
   Number of public wires:           2
   Number of public wire bits:      32
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  0
```

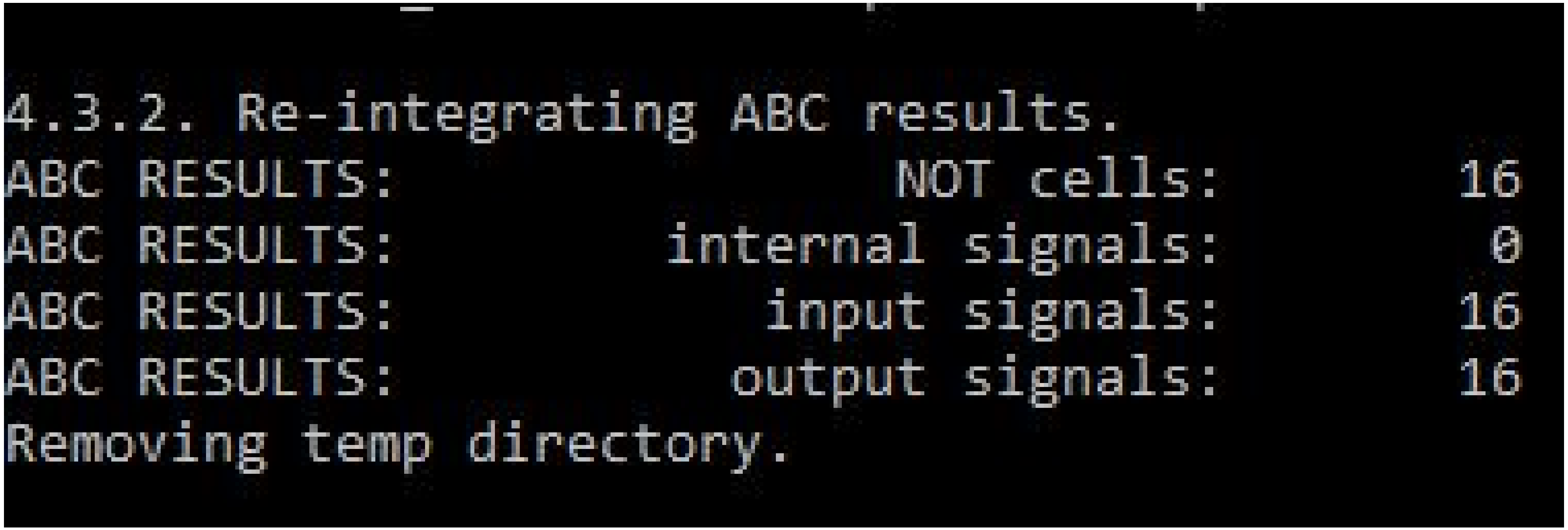

```
=== design hierarchy ===

   MyALU                              1
     ADDER                            1
     BITAND                           1
     BITINV                           1
     BITOR                            1
     MUX2                             1
     MUX4                             2
     SHIFTER_RIGHT_ARTH               2

   Number of wires:                 283
   Number of wire bits:             870
   Number of public wires:           52
   Number of public wire bits:      639
   Number of memories:                0
   Number of memory bits:             0
   Number of processes:               0
   Number of cells:                 347
     $_AND_                          31
     $_AOI3_                         12
     $_MUX_                         112
     $_NAND_                         21
     $_NOR_                          40
     $_NOT_                          57
     $_OAI3_                          8
     $_OR_                           33
     $_XNOR_                         16
     $_XOR_                          17
```



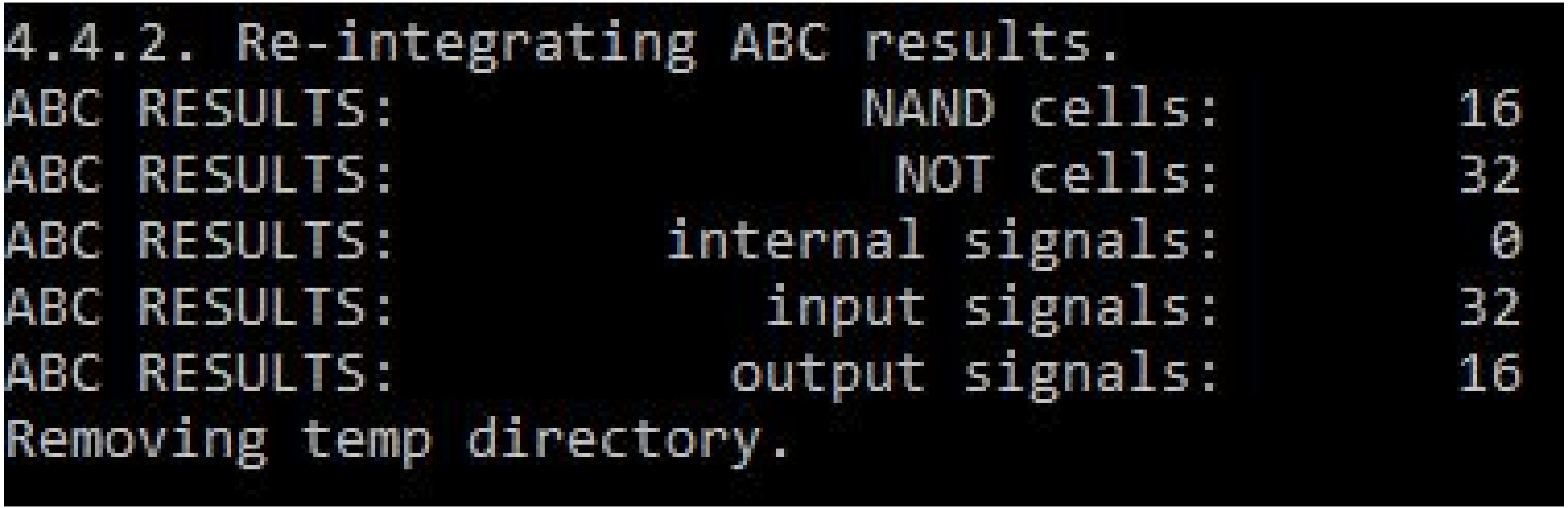
```
4.1.2. Re-integrating ABC results.
ABC RESULTS:               NAND cells:      37
ABC RESULTS:                NOR cells:     110
ABC RESULTS:                NOT cells:      48
ABC RESULTS:         internal signals:      75
ABC RESULTS:            input signals:      33
ABC RESULTS:           output signals:      16
Removing temp directory.
```



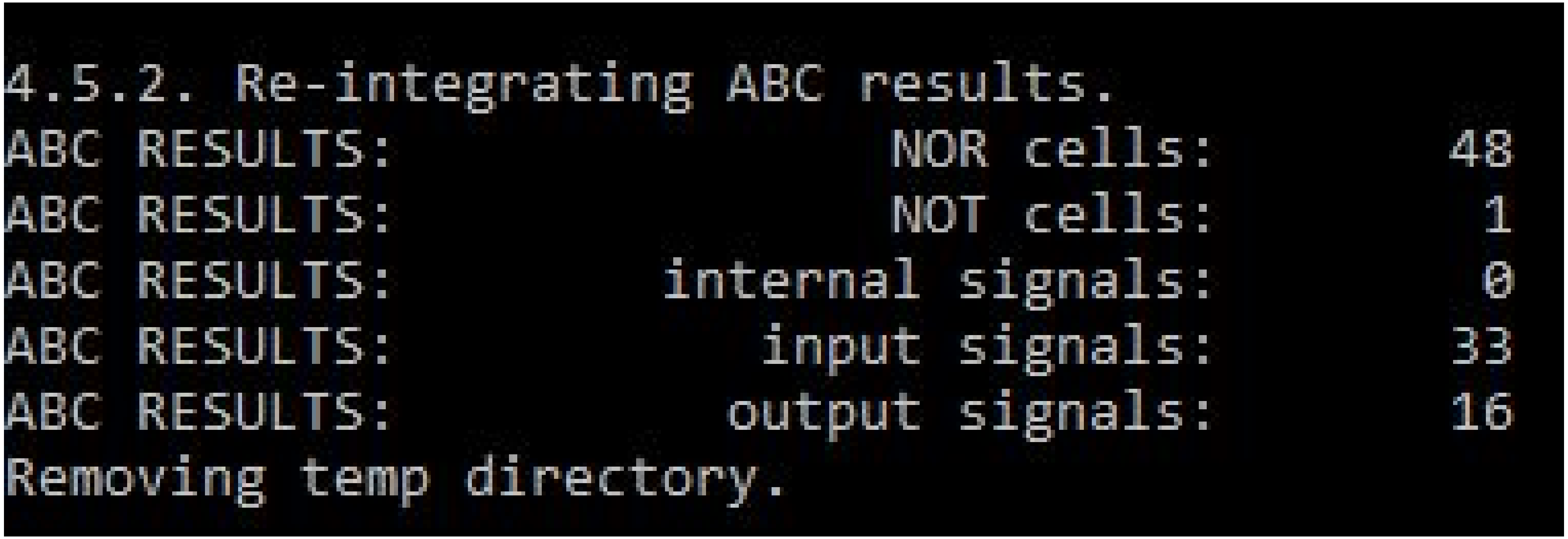
```
4.2.2. Re-integrating ABC results.
ABC RESULTS:               NOR cells:       16
ABC RESULTS:               NOT cells:       32
ABC RESULTS:        internal signals:        0
ABC RESULTS:           input signals:       32
ABC RESULTS:          output signals:       16
Removing temp directory.
```



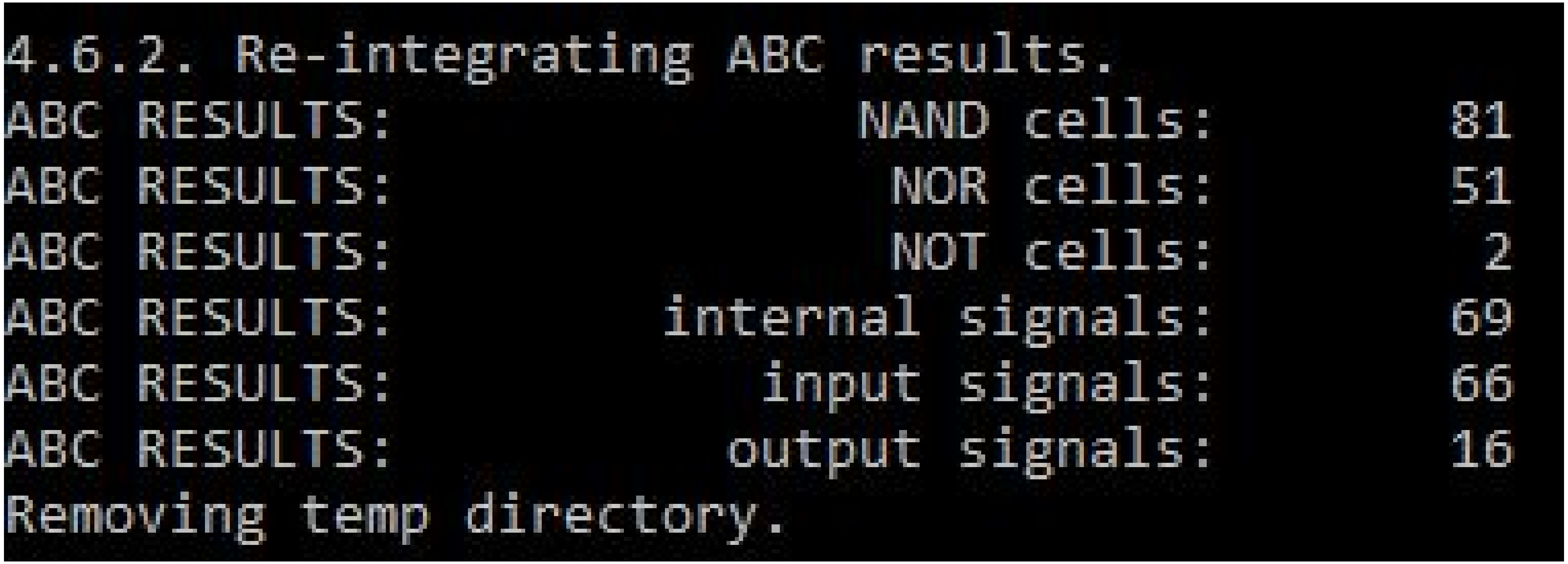
```
4.3.2. Re-integrating ABC results.
ABC RESULTS:               NOT cells:        16
ABC RESULTS:        internal signals:         0
ABC RESULTS:           input signals:        16
ABC RESULTS:          output signals:        16
Removing temp directory.
```



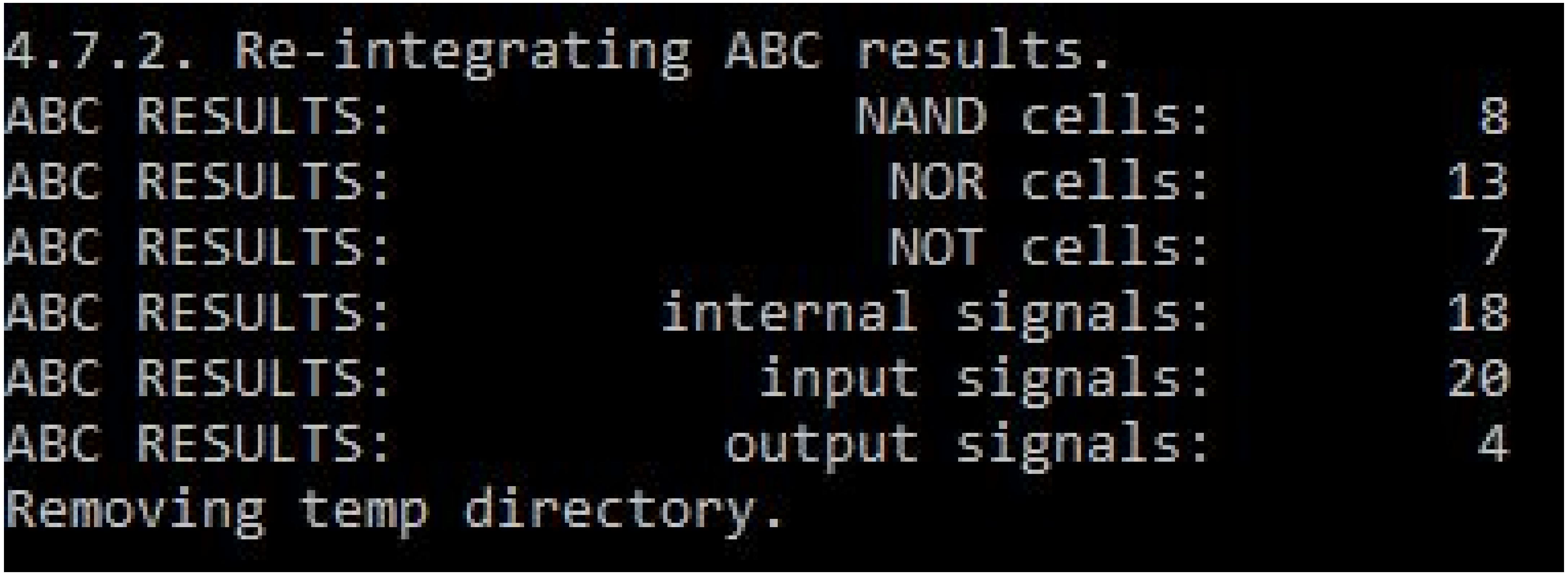
```
4.4.2. Re-integrating ABC results.
ABC RESULTS:                   NAND cells:        16
ABC RESULTS:                    NOT cells:        32
ABC RESULTS:             internal signals:         0
ABC RESULTS:                input signals:        32
ABC RESULTS:               output signals:        16
Removing temp directory.
```



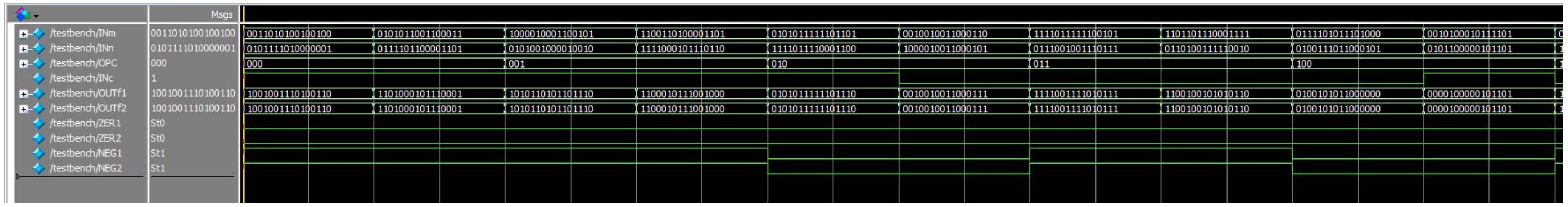
```
4.5.2. Re-integrating ABC results.
ABC RESULTS:                    NOR cells:          48
ABC RESULTS:                    NOT cells:           1
ABC RESULTS:             internal signals:           0
ABC RESULTS:                input signals:          33
ABC RESULTS:               output signals:          16
Removing temp directory.
```



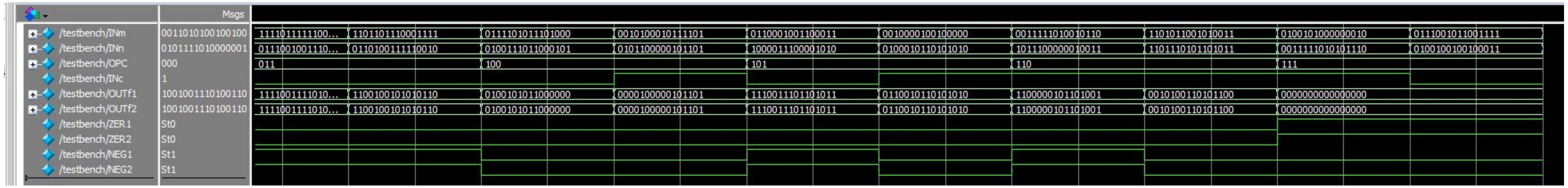
```
4.6.2. Re-integrating ABC results.
ABC RESULTS:                NAND cells:       81
ABC RESULTS:                 NOR cells:       51
ABC RESULTS:                 NOT cells:        2
ABC RESULTS:          internal signals:       69
ABC RESULTS:             input signals:       66
ABC RESULTS:            output signals:       16
Removing temp directory.
```



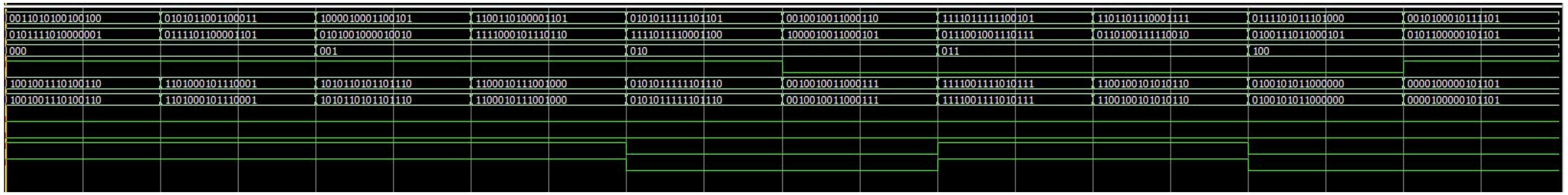
```
4.7.2. Re-integrating ABC results.
ABC RESULTS:                   NAND cells:      8
ABC RESULTS:                    NOR cells:     13
ABC RESULTS:                    NOT cells:      7
ABC RESULTS:             internal signals:     18
ABC RESULTS:                input signals:     20
ABC RESULTS:               output signals:      4
Removing temp directory.
```



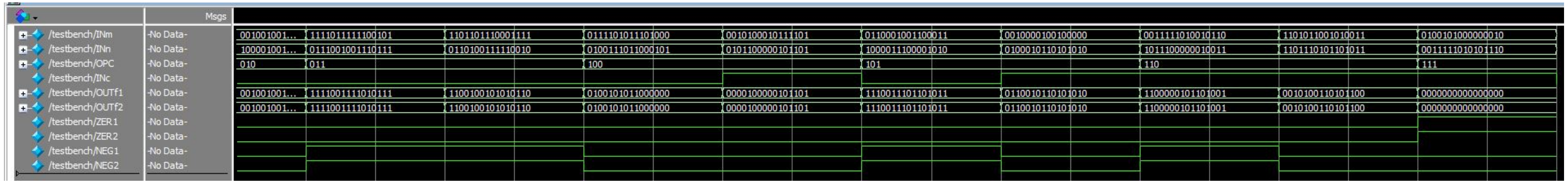
```
Msgs
/testbench/INm     0011010100100100  0011010100100100  0101011001100011  1000010001100101  1100110100001101  0101011111101101  0010010011000110  1111011111100101  1101101110001111  0111101011101000  0010100010111101
/testbench/INn     0101111010000001  0101111010000001  0111101100001101  0101001000010010  1111000101110110  1111011110001100  1000010011000101  0111001001110111  0110100111110010  0100111011000101  0101100000101101
/testbench/OPC     000               000                                 001                                 010                                 011                                 100
/testbench/INc     1
/testbench/OUTf1   1001001110100110  1001001110100110  1101000101110001  1010110101101110  1100010111001000  0101011111101110  0010010011000111  1111001111010111  1100100101010110  0100101011000000  0000100000101101
/testbench/OUTf2   1001001110100110  1001001110100110  1101000101110001  1010110101101110  1100010111001000  0101011111101110  0010010011000111  1111001111010111  1100100101010110  0100101011000000  0000100000101101
/testbench/ZER1    St0
/testbench/ZER2    St0
/testbench/NEG1    St1
/testbench/NEG2    St1
```



```
                     Msgs
/testbench/INm       0011010100100100   1111011111100...  1101101110001111  0111101011101000  0010100010111101  0110001001100011  0010000100100000  0011111010010110  1101011001010011  0100101000000010  0111001011001111
/testbench/INn       0101111010000001   0111001001110...  0110100111110010  0100111011000101  0101100000101101  1000011100001010  0100010110101010  1011100000010011  1101110101101011  0011111010101110  0100100100100011
/testbench/OPC       000                011                                 100                                 101                                 110                                 111
/testbench/INc       1
/testbench/OUTf1     1001001110100110   1111001111010...  1100100101010110  0100101011000000  0000100000101101  1110011101101011  0110010110101010  1100000101101001  0010100110101100  0000000000000000
/testbench/OUTf2     1001001110100110   1111001111010...  1100100101010110  0100101011000000  0000100000101101  1110011101101011  0110010110101010  1100000101101001  0010100110101100  0000000000000000
/testbench/ZER1      St0
/testbench/ZER2      St0
/testbench/NEG1      St1
/testbench/NEG2      St1
```



```
0011010100100100  0101011001100011  1000010001100101  1100110100001101  0101011111101101  0010010011000110  1111011111100101  1101101110001111  0111101011101000  0010100010111101
0101111010000001  0111101100001101  0101001000010010  1111000101110110  1111011110001100  1000010011000101  0111001001110111  0110100111110010  0100111011000101  0101100000101101
000                                 001                                 010                                 011                                 100
1001001110100110  1101000101110001  1010110101101110  1100010111001000  0101011111101110  0010010011000111  1111001111010111  1100100101010110  0100101011000000  0000100000101101
1001001110100110  1101000101110001  1010110101101110  1100010111001000  0101011111101110  0010010011000111  1111001111010111  1100100101010110  0100101011000000  0000100000101101
```


```
                      Msgs
/testbench/INm    -No Data-  001001001... 1111011111100101 1101101110001111 0111101011101000 0010100010111101 0110001001100011 0010000100100000 0011111010010110 1101011001010011 0100101000000010
/testbench/INn    -No Data-  100001001... 0111001001110111 0110100111110010 0100111011000101 0101100000101101 1000011100001010 0100010110101010 1011100000010011 1101110101101011 0011111010101110
/testbench/OPC    -No Data-  010          011                               100                               101                               110                               111
/testbench/INc    -No Data-
/testbench/OUTf1  -No Data-  001001001... 1111001111010111 1100100101010110 0100101011000000 0000100000101101 1110011101101011 0110010110101010 1100000101101001 0010100110101100 0000000000000000
/testbench/OUTf2  -No Data-  001001001... 1111001111010111 1100100101010110 0100101011000000 0000100000101101 1110011101101011 0110010110101010 1100000101101001 0010100110101100 0000000000000000
/testbench/ZER1   -No Data-
/testbench/ZER2   -No Data-
/testbench/NEG1   -No Data-
/testbench/NEG2   -No Data-
```