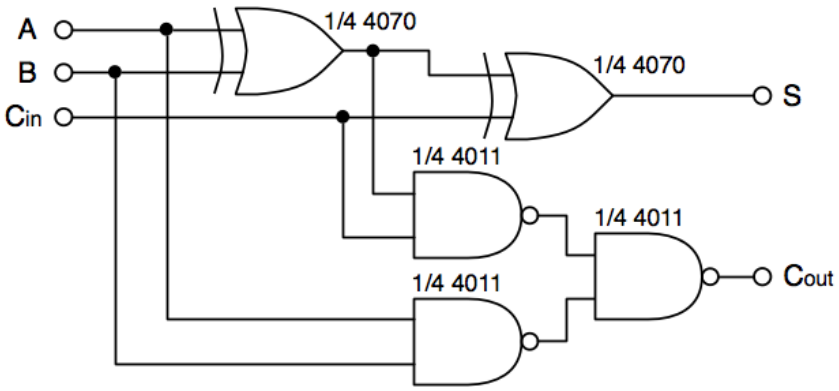
Digital Systems 1 – Computer Assignment 3
Samira Hajizadeh - 810198378

Question 1

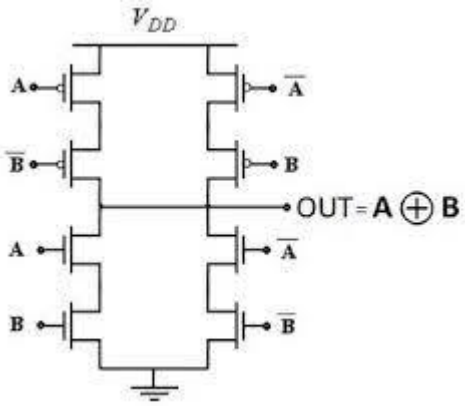
Full adder structure:

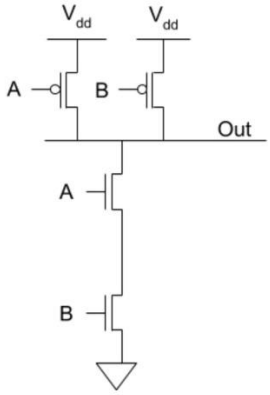


Delays:

nmos delays: (3, 4, 5)

pmos delays: (5, 6, 7)





- Inverter gate: (5, 7)
- xor gate: (17, 19) (A=B=0 to B=1, A=1 B=0 to A=0)
- nand gate: (10, 8) (A=B=1 to B=0, A=1 B=0 to B=1)

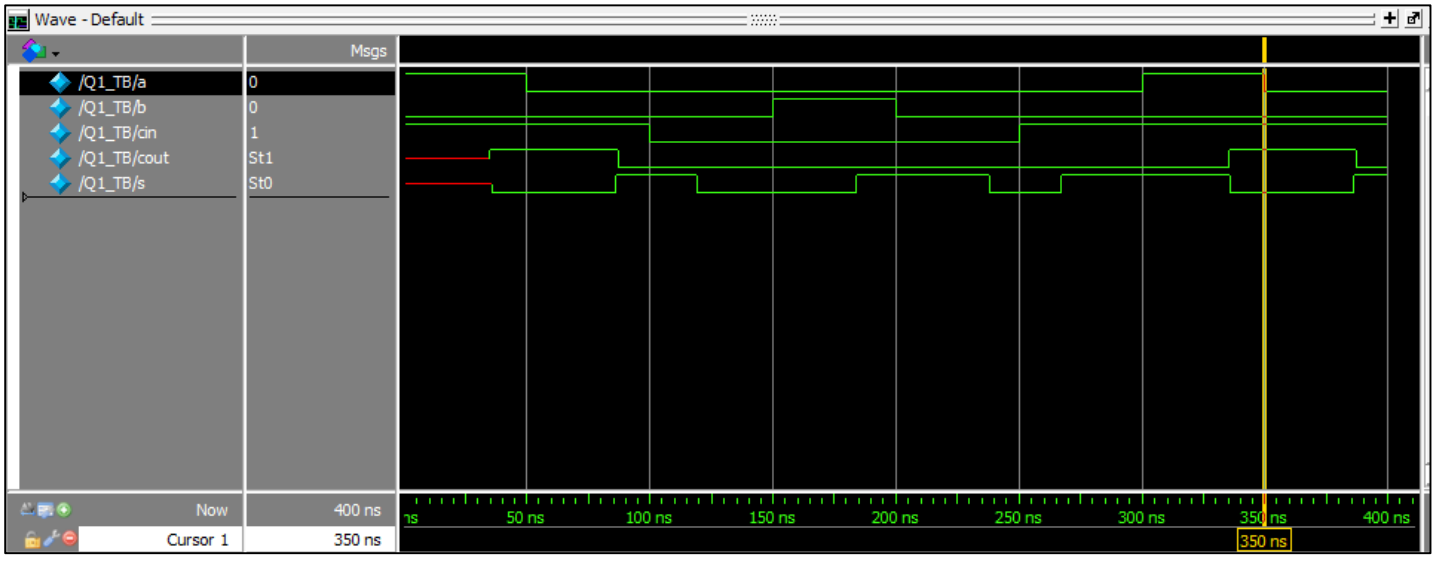
overall worst case delay:

- Sum: (36, 38) (B=0 A=Cin=1 to A=0, A=Cin=0 B=1 to B=0)
- Cout: (35, 37) (A=B=0 Cin=1 to A=1, A=Cin=1 B=0 to A=0)

Digital Systems 1 – Computer Assignment 3
Samira Hajizadeh - 810198378

```
Ln#
 1   `timescale 1ns/1ns
 2
 3   module FA(input A, B, Cin, output S, Cout);
 4
 5           wire y1, y2, y3;
 6           xor #(17, 19) g1(y1, A, B), g2(S, y1, Cin);
 7           nand #(10, 8) g3(y2, A, B), g4(y3, Cin, y1), g5(Cout, y3, y2);
 8
 9   endmodule
10
```

```
Ln#
 1   `timescale 1ns/1ns
 2
 3   module Q1_TB();
 4
 5           logic a=1, b=0, cin=1;
 6           wire cout, s;
 7
 8           FA full_adder(a, b, cin, s, cout);
 9
10           initial begin
11           #50 a = 0;
12           #50 cin = 0;
13           #50 b = 1;
14           #50 b = 0;
15           #50 cin = 1;
16           #50 a = 1;
17           #50 a = 0;
18           #50 $stop;
19           end
20
21   endmodule
22
```



All the delays are verified using the Verilog code above.

Digital Systems 1 – Computer Assignment 3
Samira Hajizadeh - 810198378

Question 2

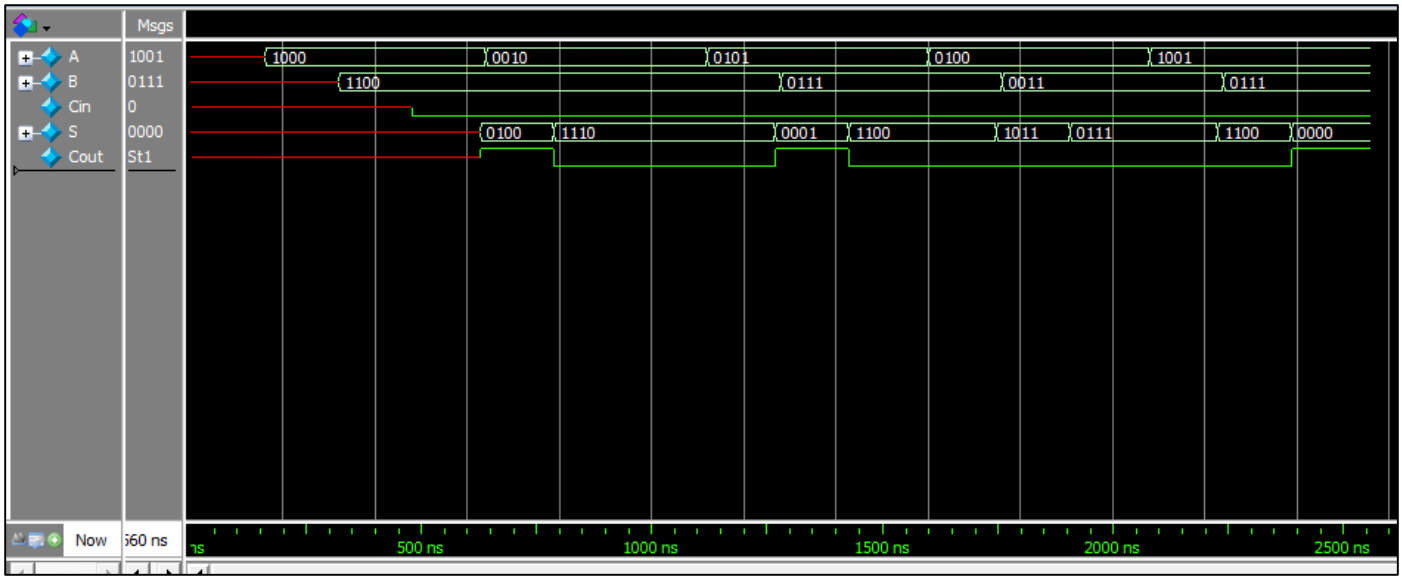
For calculating the worst case delay of a n-bit ripple-carry adder, we add the worst case delay of n-1 Couts in addition to the worst case delay of the last Summation (which is longer than the worst case delay of the last Cout). Therefore: delay = (n-1) * 37 + 38

```
Ln#
1   `timescale 1ns/1ns
2
3   module FA_nbit #(parameter n=3) (input [n-1:0] A, B, input Cin, output [n-1:0] S, output Cout);
4
5           assign #((n-1)*37 + 38) {Cout, S} = A + B + Cin;
6
7   endmodule
8
```

Question 3

```
Ln#
1   `timescale 1ns/1ns
2
3   module FA_nbit_TB();
4
5           logic [3:0] A;
6           logic [3:0] B;
7           logic Cin;
8           wire [3:0] S;
9           wire Cout;
10          parameter n = 4;
11
12          FA_nbit #(n) fa(A, B, Cin, S, Cout);
13
14          initial begin
15
16          repeat(5)
17          begin
18          #160 A = {$random} % (4'b1111);
19          #160 B = {$random} % (4'b1111);
20          #160 Cin = {$random} % (1'b1);
21          end
22
23          #160 $stop;
24          end
25
26  endmodule
27
```

Digital Systems 1 – Computer Assignment 3
Samira Hajizadeh - 810198378



Question 4

```
Ln#
 1  `timescale 1ns/1ns
 2
 3  module ones_counter_7bit (input [6:0] A, output [2:0] N);
 4
 5          wire [3:0] in_2bit;
 6          parameter n=1;
 7          genvar i;
 8
 9          generate
10          for (i=0; i < 2; i = i + 1) begin: FAs
11                  FA_nbit #(n) fa(A[i*3], A[i*3+1], A[i*3+2], in_2bit[i*2], in_2bit[i*2+1]);
12          end
13          endgenerate
14          parameter m = 2;
15          FA_nbit #(m) fa(in_2bit[1:0], in_2bit[3:2], A[6], N[1:0], N[2]);
16
17  endmodule
18
```

Digital Systems 1 – Computer Assignment 3
Samira Hajizadeh - 810198378

```
Ln#
 1  `timescale 1ns/1ns
 2
 3  module ones_counter_15bit (input [14:0] A, output [3:0] N);
 4
 5          wire [5:0] in_3bit;
 6          genvar i;
 7
 8          generate
 9          for (i=0; i < 2; i = i + 1) begin: onecounters
10                  ones_counter_7bit counter(A[i*7+6:i*7], in_3bit[i*3+2:i*3]);
11          end
12          endgenerate
13
14
15          parameter m = 3;
16          FA_nbit #(m) fa(in_3bit[2:0], in_3bit[5:3], A[14], N[2:0], N[3]);
17
18  endmodule
19
```

```
Ln#
 1  `timescale 1ns/1ns
 2
 3  module ones_counter_31bit (input [30:0] A, output [4:0] N);
 4
 5          wire [7:0] in_4bit;
 6          genvar i;
 7
 8          generate
 9          for (i=0; i < 2; i = i + 1) begin: onecounters
10                  ones_counter_15bit counter(A[i*15+14:i*15], in_4bit[i*4+3:i*4]);
11          end
12          endgenerate
13
14
15          parameter m = 4;
16          FA_nbit #(m) fa(in_4bit[3:0], in_4bit[7:4], A[30], N[3:0], N[4]);
17
18  endmodule
19
```

```
Ln#
 1  `timescale 1ns/1ns
 2
 3  module ones_counter_63bit (input [62:0] A, output [5:0] N);
 4
 5          wire [9:0] in_5bit;
 6          genvar i;
 7
 8          generate
 9          for (i=0; i < 2; i = i + 1) begin: onecounters
10                  ones_counter_31bit counter(A[i*31+30:i*31], in_5bit[i*5+4:i*5]);
11          end
12          endgenerate
13
14
15          parameter m = 5;
16          FA_nbit #(m) fa(in_5bit[4:0], in_5bit[9:5], A[62], N[4:0], N[5]);
17
18  endmodule
19
```

Digital Systems 1 – Computer Assignment 3
Samira Hajizadeh - 810198378

```
Ln#
 1  `timescale 1ns/1ns
 2
 3  module ones_counter_127bit (input [126:0] A, output [6:0] N);
 4
 5          wire [11:0] in_6bit;
 6          genvar i;
 7
 8          generate
 9          for (i=0; i < 2; i = i + 1) begin: onecounters
10                  ones_counter_63bit counter(A[i*63+62:i*63], in_6bit[i*6+5:i*6]);
11          end
12          endgenerate
13
14
15          parameter m = 6;
16          FA_nbit #(m) fa(in_6bit[5:0], in_6bit[11:6], A[126], N[5:0], N[6]);
17
18  endmodule
19
```

Question 5

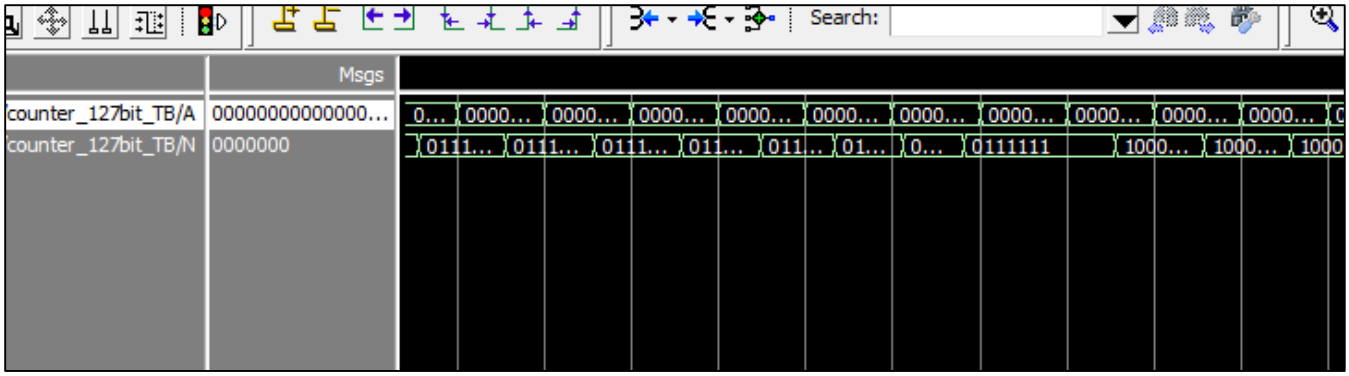
Test bench that produces 127-bit marching-1 data that covers all the bits:

```
Ln#
 1  `timescale 1ns/1ns
 2  module counter_127bit_TB ();
 3
 4          logic [126:0] A = 1'd0;
 5          wire [6:0] N;
 6
 7          ones_counter_127bit counter(A, N);
 8
 9          initial begin
10          repeat(127)
11                  begin
12                  #1000 A = 2*A + 1;
13                  end
14          repeat(127)
15                  begin
16                  #1000 A = A*2;
17                  end
18          #1000 $stop;
19          end
20
21  endmodule
22
```

Picture of the waveform:

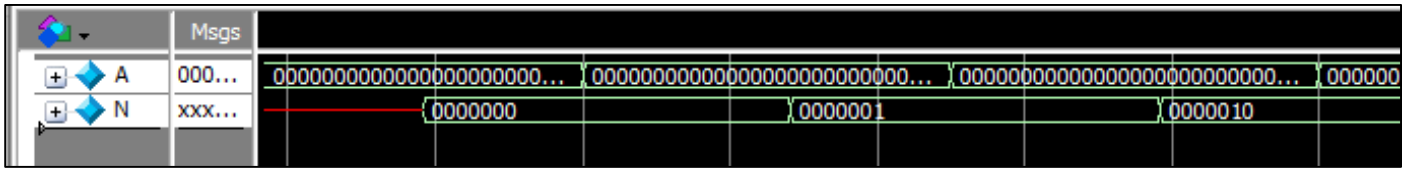
Digital Systems 1 – Computer Assignment 3

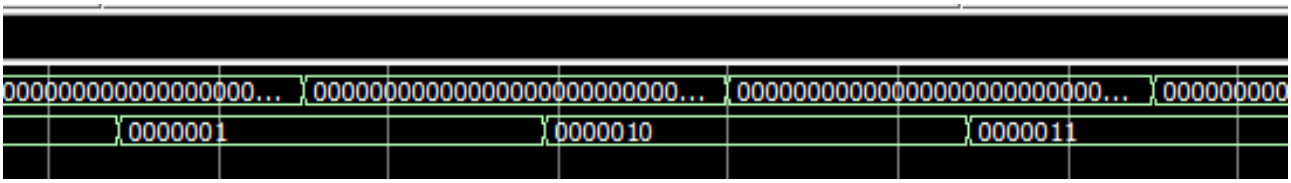
Samira Hajizadeh - 810198378



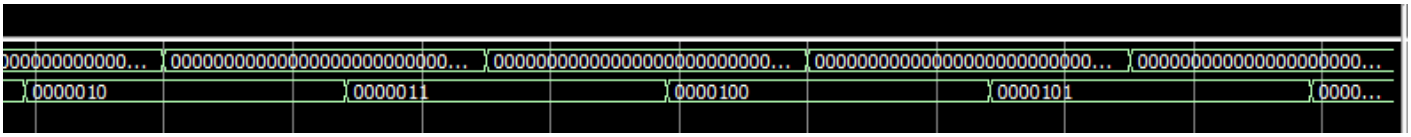
Waveform produced by this bench is too large to prove anything on a screenshot. Therefore, instead of that, this bench is used:

```
Ln#
 1   `timescale 1ns/1ns
 2   module counter_127bit_TB_final ();
 3
 4           logic [126:0] A = 1'd0;
 5           wire [6:0] N;
 6
 7           ones_counter_127bit counter(A, N);
 8           initial begin
 9           repeat(8)
10                   begin
11                   #1000 A = 2*A + 1;
12                   end
13           repeat(10)
14                   begin
15                   #1000 A = A*2;
16                   end
17           repeat(5)
18                   begin
19                   #1000 A = {$random} % (1'd1);
20                   end
21           #1000 $stop;
22           end
23
24   endmodule
```





Digital Systems 1 – Computer Assignment 3
Samira Hajizadeh - 810198378



Again only some parts of the waveform are shown, due to its length.

Question 6

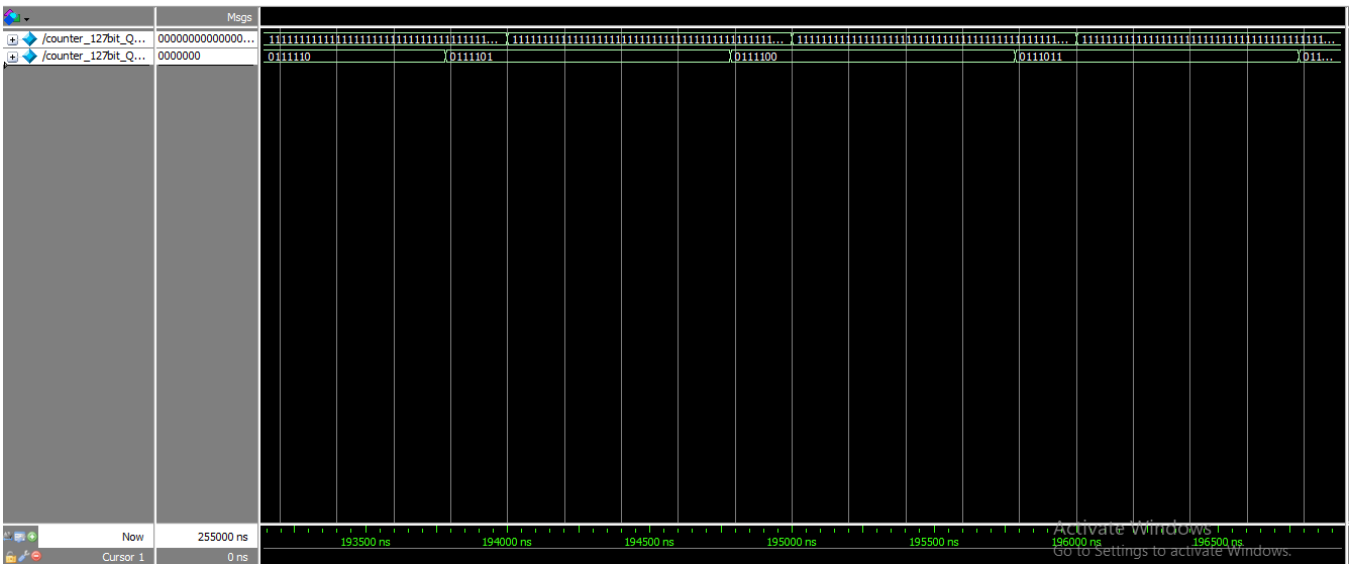
Delay of the one's counter of part 4:

38 * 6 + 37 * (0+1+2+3+4+5) = 783 = 19783 – 19000


```
Ln#
 1  `timescale 1ns/1ns
 2
 3  module ones_counter_127bit_Q6 (input [126:0] A, output reg [6:0] N);
 4          integer i;
 5          always @(A) begin
 6                  #783 N = 7'b0000000;
 7                  for (i=0; i<127; i=i+1) begin
 8                          if (A[i]==1'b1)
 9                                  N=N+(1'b1);
10                  end
11          end
12 endmodule
13
```

Digital Systems 1 – Computer Assignment 3
Samira Hajizadeh - 810198378

```
Ln#
 1   `timescale 1ns/1ns
 2   module counter_127bit_Q6_TB ();
 3
 4           logic [126:0] A = 1'd0;
 5           wire [6:0] N;
 6
 7           ones_counter_127bit_Q6 counter(A, N);
 8
 9           initial begin
10           repeat(127)
11                   begin
12                   #1000 A = 2*A + 1;
13                   end
14           repeat(127)
15                   begin
16                   #1000 A = A*2;
17                   end
18           #1000 $stop;
19           end
20
21   endmodule
22
```



Due to similar delays the results of the stimulations are the same.

Question 7

Q4's one's counter:

Digital Systems 1 – Computer Assignment 3

Samira Hajizadeh - 810198378

```
=== design hierarchy ===

   ones_counter_127bit              1
     $paramod\FA_nbit\n=6           1
     ones_counter_63bit             2
       $paramod\FA_nbit\n=5         1
       ones_counter_31bit           2
         $paramod\FA_nbit\n=4       1
         ones_counter_15bit         2
           $paramod\FA_nbit\n=3     1
           ones_counter_7bit        2
             $paramod\FA_nbit\n=1     2
             $paramod\FA_nbit\n=2     1

   Number of wires:                777
   Number of wire bits:           1759
   Number of public wires:         408
   Number of public wire bits:    1390
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                552
     $_AND_                         46
     $_AOI3_                        46
     $_NAND_                        83
     $_NOR_                          3
     $_NOT_                         46
     $_OAI3_                        82
     $_OR_                           6
     $_XNOR_                       128
     $_XOR_                        112
```

```
4.6.2. Re-integrating ABC results.
ABC RESULTS:               NAND cells:       18
ABC RESULTS:                NOR cells:       39
ABC RESULTS:                NOT cells:       19
ABC RESULTS:        internal signals:        25
ABC RESULTS:           input signals:        13
ABC RESULTS:          output signals:         7
Removing temp directory.
```

Q6’s one’s counter:

Digital Systems 1 – Computer Assignment 3
Samira Hajizadeh - 810198378

```
=== ones_counter_127bit_Q6 ===

   Number of wires:               3189
   Number of wire bits:           3321
   Number of public wires:           2
   Number of public wire bits:     134
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:               3194
     $_AND_                        244
     $_AOI3_                       592
     $_MUX_                        496
     $_NAND_                       134
     $_NOR_                        837
     $_NOT_                         13
     $_OAI3_                         9
     $_OR_                           9
     $_XNOR_                       620
     $_XOR_                        240

2.24. Executing CHECK pass (checking for obvious problems).
checking module ones_counter_127bit_Q6..
found and reported 0 problems.
```

```
4.1.2. Re-integrating ABC results.
ABC RESULTS:               NAND cells:     1314
ABC RESULTS:                NOR cells:     1795
ABC RESULTS:                NOT cells:      536
ABC RESULTS:        internal signals:     3187
ABC RESULTS:           input signals:      127
ABC RESULTS:          output signals:        7
Removing temp directory.
```

The result of the second module uses way more gates and leads to a higher power consumption due to the fact that the yosys's path is not always the optimized one. In the first case, because we elaborated the design for yosys the result uses ten times less gates.