به نام خدا

UNIVERSITY OF TEHRAN
Electrical and Computer Engineering Department

Computer Assignment 3

Digital Systems I - ECE 894

Deadline : 25 Ordibehesht 1400

Erfan Panahi

810198369

Spring 1400

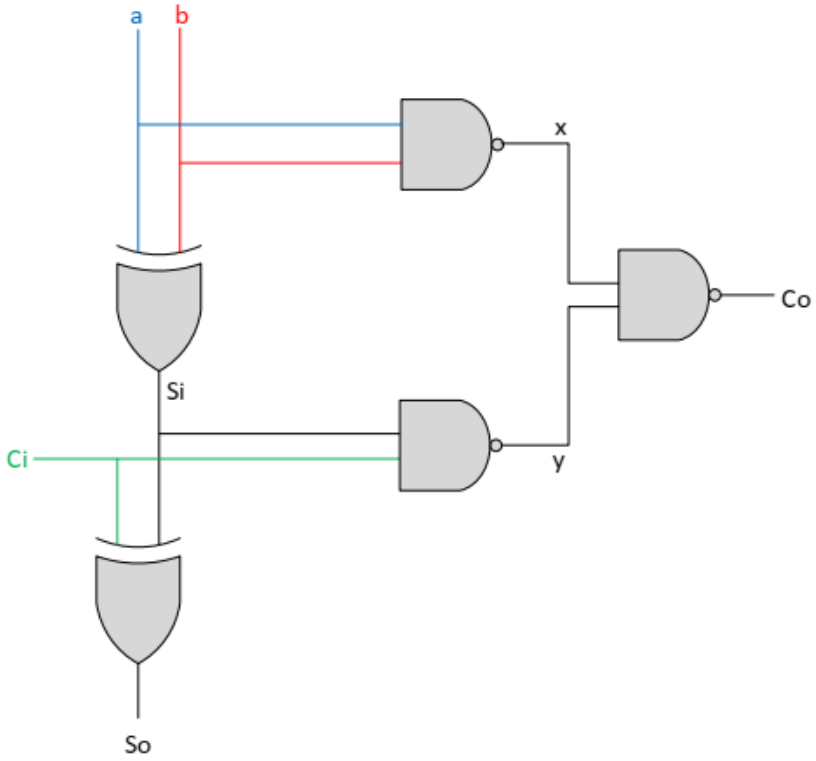
List of Problems:.

Computer Assignment 2

Problem 1. File: Fadder_P1.sv and Fadder_P1_TB.sv

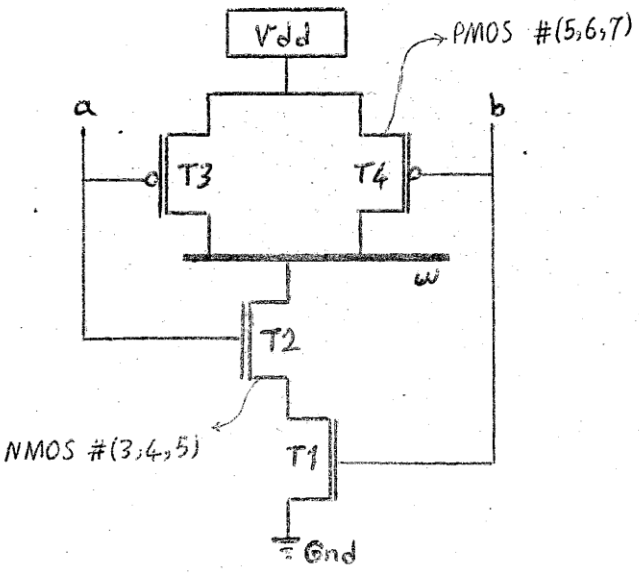
A. Circuit diagram:

In following figure we show the design of a full-adder using basic gate structure:

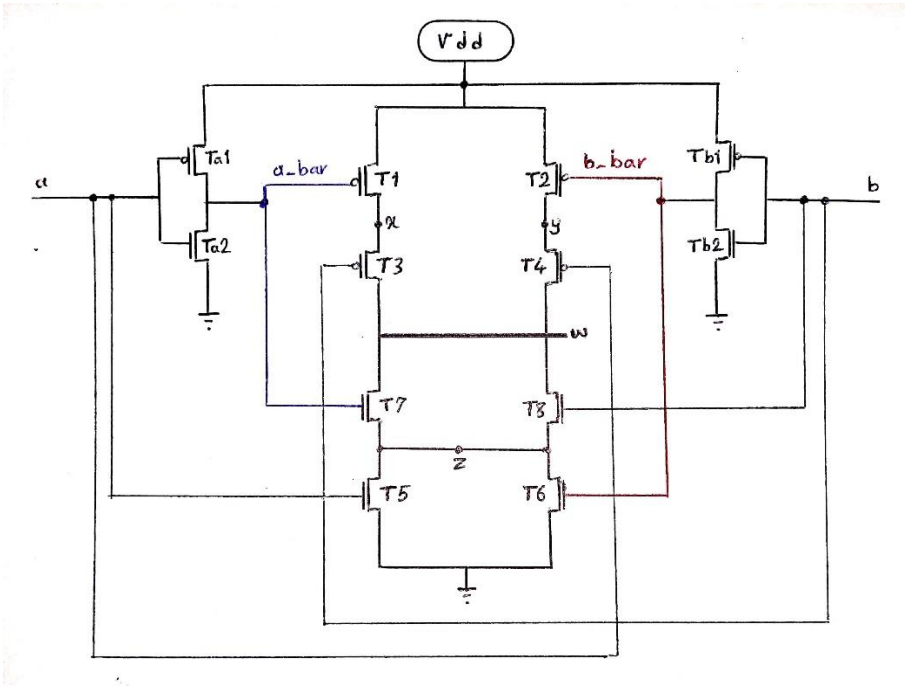


We have used XOR and NAND gates in this structure. So, we show the transistor level structure of this gates.

NAND gate structure:



XOR gate structure:



B. Worst-cases:

NAND gate :

We have calculated the worst case of the NAND gate in CA1:

Worst case To1 : 10NS $(ab = 11 \rightarrow 10)$

Worst case To0 : 8NS $(ab = 10 \rightarrow 11)$

Using 'nand' statement:

```
nand #(10,8) (w,a,b);
```

XOR gate :

We have calculated the worst case of the NAND gate in HW2:

Worst case To1 : 17NS $(ab = 11 \rightarrow 10)$

Worst case To0 : 19NS $(ab = 10 \rightarrow 11)$

Using 'assign' statement: (Because SystemVerilog does not have XOR statement)

```
assign #(17,19) w = a ^ b;
```

Full-adder :

For So (Sum Out):

Worst case To1 : 19 + 17 = 36NS $(abc_i = 101 \rightarrow 001)$

Worst case To0 : 17 + 19 = 36NS $(abc_i = 001 \rightarrow 101)$

For Co (Carry Out):

Worst case To1 : 17 + 8 + 10 = 35NS $(abc_i = 001 \rightarrow 101)$

Worst case To0 : 19 + 10 + 8 = 37NS $(abc_i = 101 \rightarrow 001)$

C. System Verilog Description:

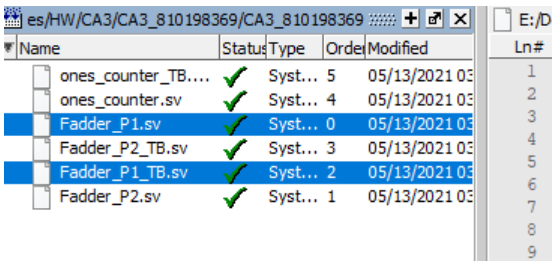
File: Fadder_P1.sv

```
`timescale 1ns/1ns
module Fadder_P1(input a,b,ci,output so,co);
      wire si,x,y;
      assign #(17,19) si = a ^ b;
      assign #(17,19) so = si ^ ci;
      nand #(10,8) (x,a,b);
      nand #(10,8) (y,si,ci);
      nand #(10,8) (co,x,y);
endmodule
```

File: Fadder_P1_TB.sv

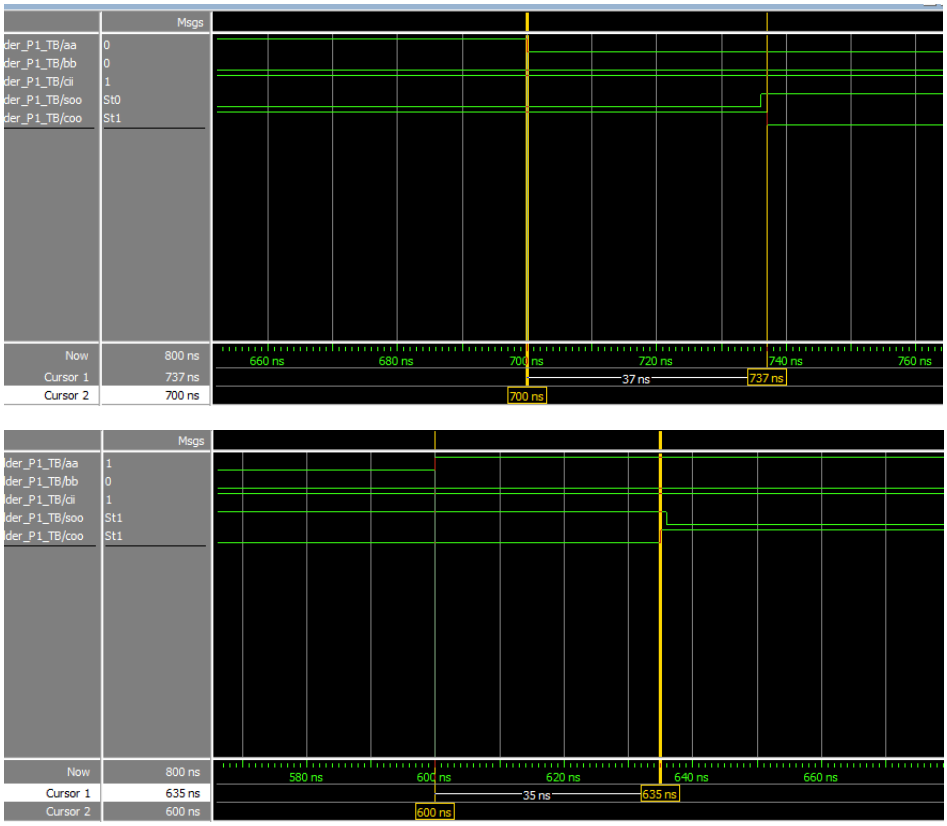
```
`timescale 1ns/1ns
module Fadder_P1_TB();
      logic aa=0,bb=0,cii=0;
      wire soo,coo;
      Fadder_P1 UUT(aa,bb,cii,soo,coo);
      initial begin
       #100 aa = 1;
       #100 bb = 1;
       #100 bb = 0;
       #100 aa = 0;
       #100 cii = 1;
       #100 aa = 1;
       #100 aa = 0;
       #100 $stop;
end
endmodule
```

D. Image of project and compiling:

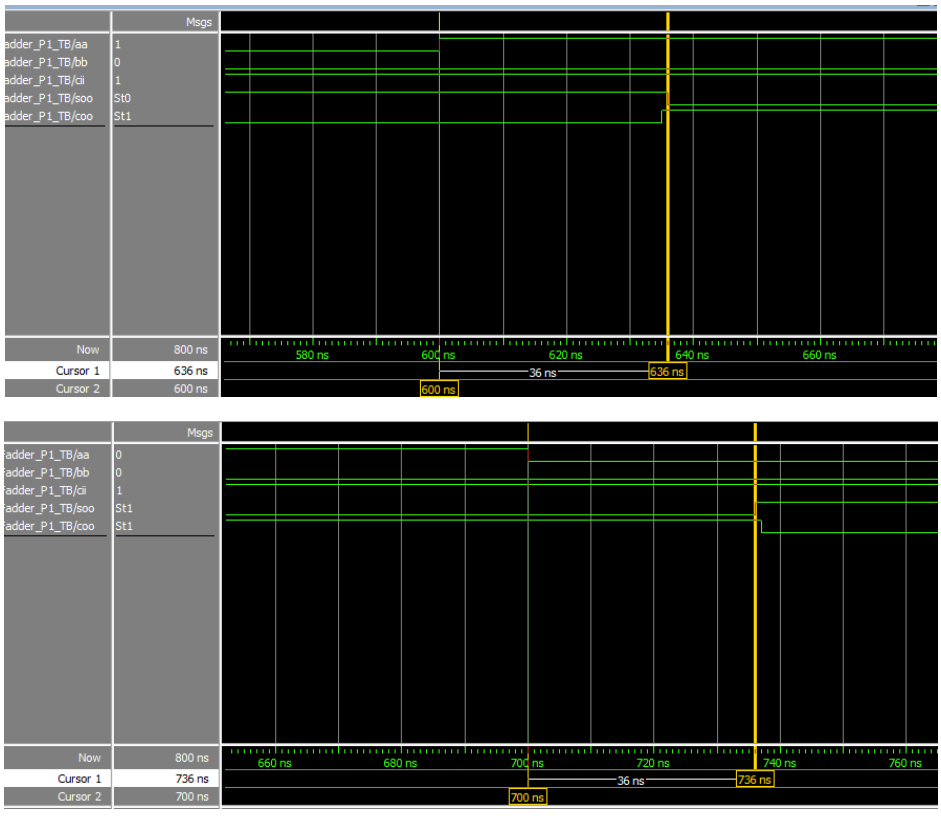


E. Waveform:

Carry Out worst-case:



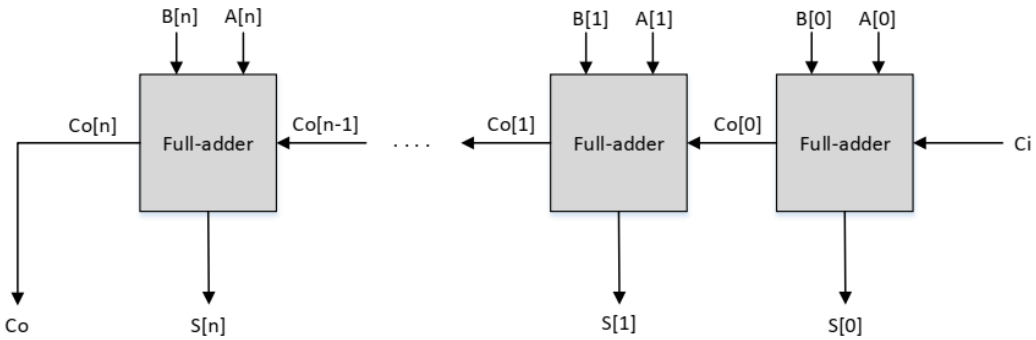
Sum Out worst-case:



Problem 2. Files: Fadder_P2.sv

A. Delay values and circuit diagram:

According to the Problem 2 of CA for the delay value we assume the adder is realized as a Ripple-carry adder. The diagram of RCA is shown here:



We consider the same worst delay for the both output that is maximum of T01:{36,37} To1:{36,35}. So we can say the worst case delay of the full-adder RTL is:

To1 = 37NS

To0 = 36NS

Using 'assign' statement we write a full-adder:

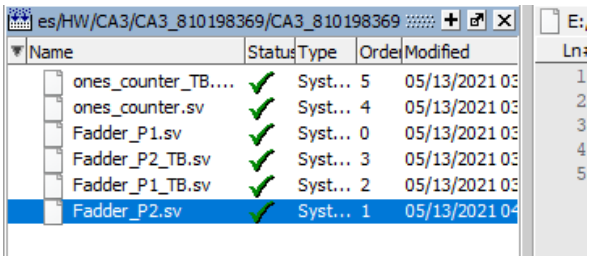
```
assign #(37,36) {co,so} =  a + b + ci;
```

B. System Verilog Description:

File: Fadder_P2.sv

```
`timescale 1ns/1ns
module Fadder_P2 #(parameter n) (input [n-1:0]A,B,input Ci,output [n-1:0]So,output Co);
      assign #(n*37,n*36) {Co,So} =  A + B + Ci;
endmodule
```

C. Image of project and compiling:



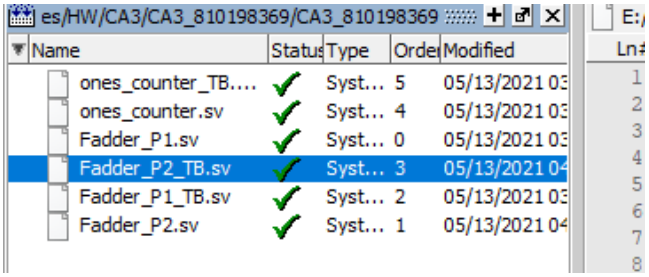
Problem 3. Files: Fadder_P2_TB.sv

A. System Verilog Description:

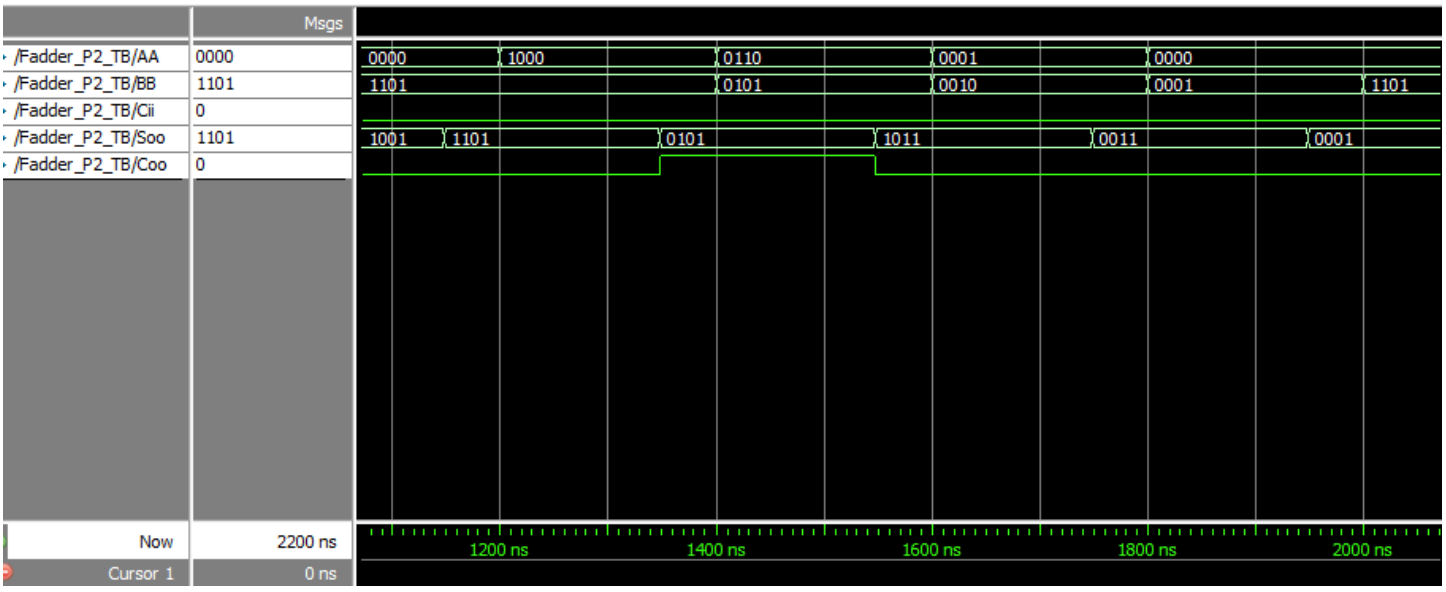
File: Fadder_P2_TB.sv

```
`timescale 1ns/1ns
module Fadder_P2_TB();
      logic [3:0]AA,BB;
      logic Cii = 0;
      wire [3:0]Soo;
      logic Coo;
      Fadder_P2 #4 UUT(AA,BB,Cii,Soo,Coo);
      initial begin
       repeat (10) #200 {AA,BB} = $random;
       #200 $stop;
      end
endmodule
```

B. Image of project and compiling:



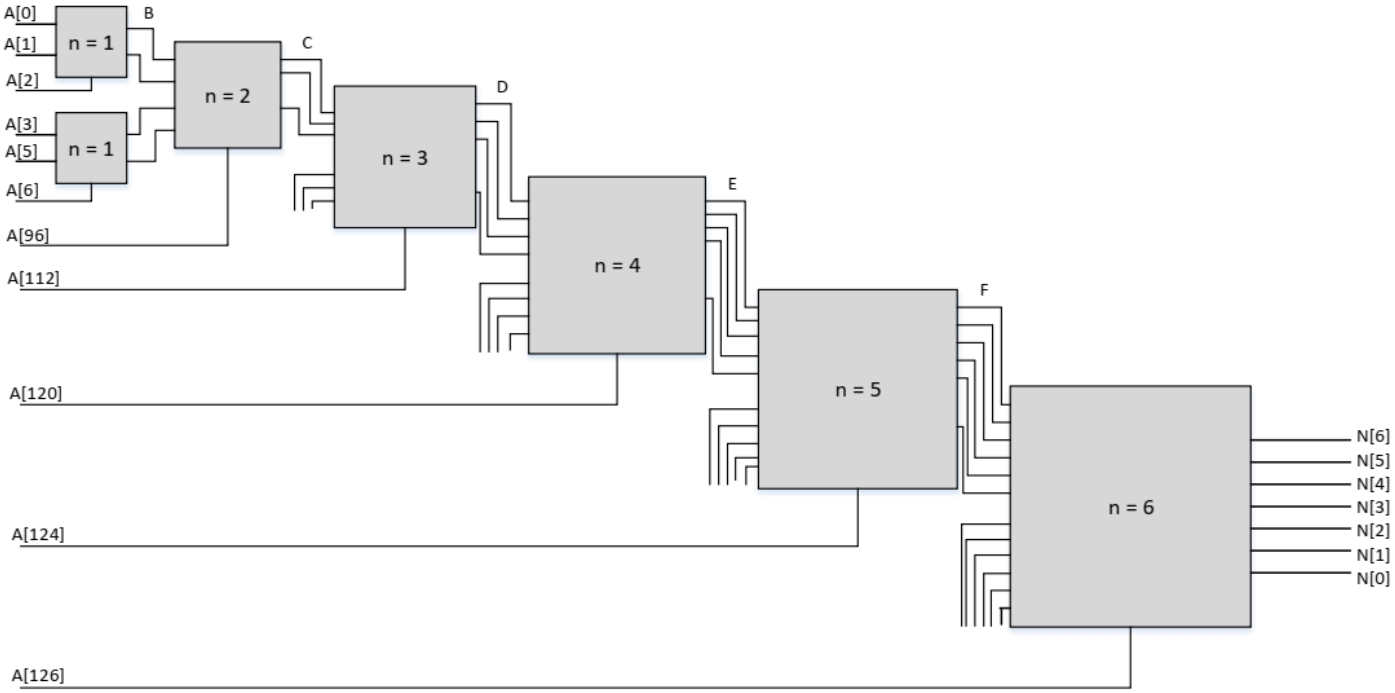
C. Waveform:



Problem 4. Files: My1s_counter.sv

A. Circuit diagram:

In following figure we show the design of a 127-bit 1's counter: (It is not complete diagram and each block shows a full-adder.)



B. Worst-case:

As we see we use n = 1-to-6 full-adders in this RTL. According to Problem 2s' delay we can say:

Worst case of 127 bit 1's counter: $(1 + 2 + 3 + 4 + 5 + 6) \times 37 = 777NS$

C. System Verilog Description:

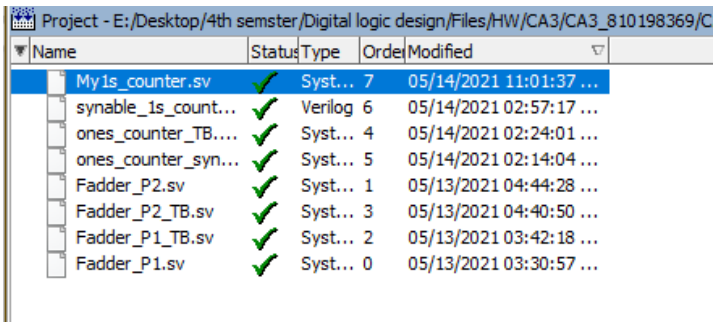
File: My1s_counter.sv

```
`timescale 1ns/1ns
module My1s_counter127(input [126:0]A,output [6:0]N);
      wire [63:0]B;
      wire [47:0]C;
      wire [31:0]D;
      wire [19:0]E;
      wire [11:0]F;
      genvar i;
      generate
```

```
        for (i=0;i<32;i=i+1) begin: Fa1bit
             Fadder #1 fa1(A[3*i+1],A[3*i],A[3*i+2],B[2*i],B[2*i+1]);
        end
        for (i=0;i<16;i=i+1) begin: Fa2bit
             Fadder #2
fa2({B[4*i+3],B[4*i+2]},{B[4*i+1],B[4*i]},A[96+i],{C[3*i+1],C[3*i]},C[3*i+2]);
        end
        for (i=0;i<8;i=i+1) begin: Fa3bit
             Fadder #3
fa3({C[6*i+5],C[6*i+4],C[6*i+3]},{C[6*i+2],C[6*i+1],C[6*i]},A[112+i],{D[4*i+2],D[4*i+1],D
[4*i]},D[4*i+3]);
        end
        for (i=0;i<4;i=i+1) begin: Fa4bit
             Fadder #4
fa4({D[8*i+7],D[8*i+6],D[8*i+5],D[8*i+4]},{D[8*i+3],D[4*i+2],D[8*i+1],D[8*i]},A[120+i],{E
[5*i+3],E[5*i+2],E[5*i+1],E[5*i]},E[5*i+4]);
        end
        for (i=0;i<2;i=i+1) begin: Fa5bit
             Fadder #5
fa5({E[10*i+9],E[10*i+8],E[10*i+7],E[10*i+6],E[10*i+5]},{E[10*i+4],E[10*i+3],E[10*i+2],E[
10*i+1],E[10*i]},A[124+i],{F[6*i+4],F[6*i+3],F[6*i+2],F[6*i+1],F[6*i]},F[6*i+5]);
        end
        for (i=0;i<1;i=i+1) begin: Fa6bit
             Fadder #6
fa6({F[12*i+11],F[12*i+10],F[12*i+9],F[12*i+8],F[12*i+7],F[12*i+6]},{F[12*i+5],F[12*i+4],
F[12*i+3],F[12*i+2],F[12*i+1],F[12*i]},A[126+i],{N[7*i+5],N[7*i+4],N[7*i+3],N[7*i+2],N[7*
i+1],N[7*i]},N[7*i+6]);
        end
      endgenerate
endmodule

module Fadder #(parameter n) (input [n-1:0]A,B,input Ci,output [n-1:0]So,output Co);
      assign #(n*37,n*36) {Co,So} =  A + B + Ci;
endmodule
```

D. Image of project and compiling:



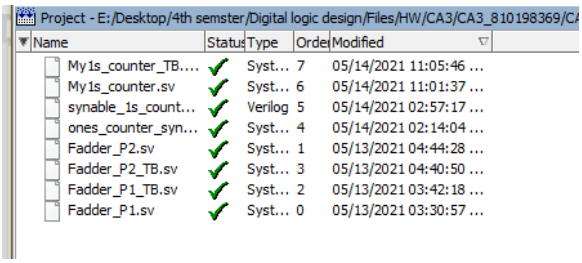
Problem 5. Files: My1s_counter _TB.sv

A. System Verilog Description:

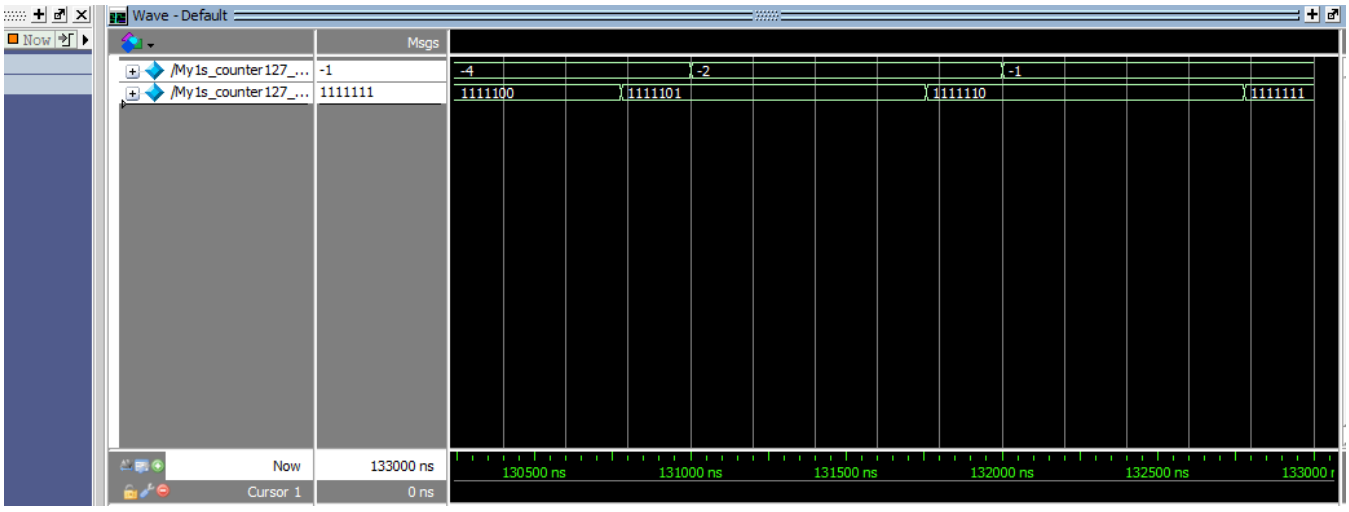
File: My1s_counter_TB.sv

```
`timescale 1ns/1ns
module My1s_counter127_TB();
logic signed [126:0]AA;
wire [6:0]NN;
My1s_counter127 UUT(AA,NN);
initial begin
AA = 127'd63;
#1000 AA = 127'd15;
#1000 AA = 127'd1;
#1000 AA = 127'd7;
#1000 AA = 127'd127;
#1000 AA = 127'd0;
#1000 AA[126] = 1'b1;
repeat (126) #1000 AA = AA >>> 1;
#1000 $stop;
end
endmodule
```

B. Image of project and compiling:



C. Waveform:



Problem 6. Files: ones_counter_synable.sv

A. System Verilog Description:

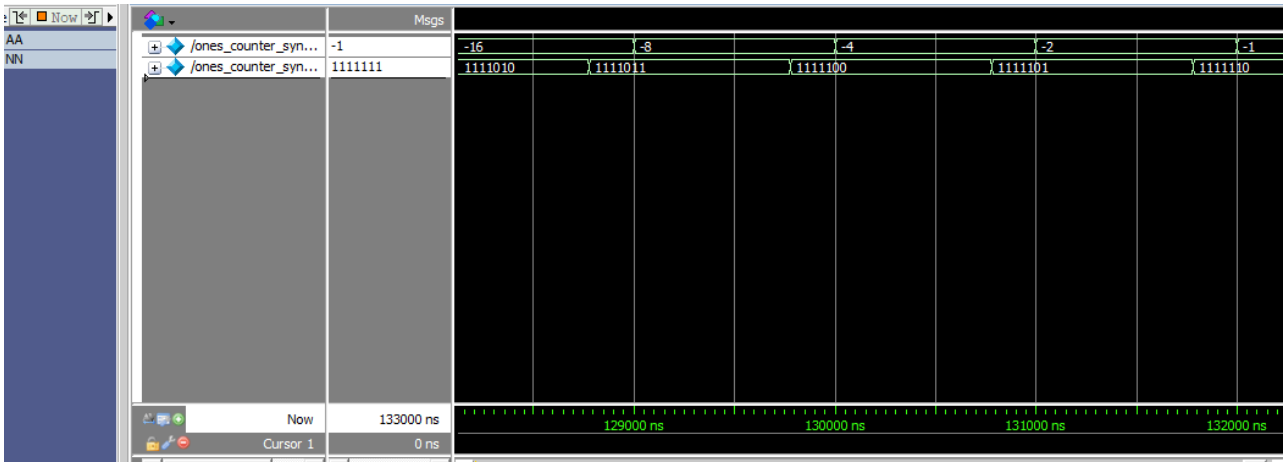
File: ones_counter_synable.sv

```
`timescale 1ns/1ns
module ones_counter_synable(input [126:0]A,output logic [6:0]N);
      int i;
      always @ (A) begin
       #777 N=7'd0;
       for (i=0;i<127;i=i+1) begin
            if (A[i] == 1'b1) N=N+1;
            end
       end
endmodule
```

Testbench: ones_counter_synable_TB.sv

```
`timescale 1ns/1ns
module ones_counter_synable_TB();
logic signed [126:0]AA;
wire [6:0]NN;
ones_counter_synable UUT(AA,NN);
initial begin
AA = 127'd63;
#1000 AA = 127'd15;
#1000 AA = 127'd1;
#1000 AA = 127'd7;
#1000 AA = 127'd127;
#1000 AA = 127'd0;
#1000 AA[126] = 1'b1;
repeat (126) #1000 AA = AA >>> 1;
#1000 $stop;
end
endmodule
```

B. Waveform: (Same result with problem 4)



Problem 7.

Part a. Synthesizing Problem 4's module:

For this part we most make Problem 4's module as a synthesizable description. So we merge the two module used in a module and make file as Verilog description (.v) .

File: My1s_counter_P4_synable.v

```
`timescale 1ns/1ns
module My1s_counter_P4_synable(input [126:0]A,output [6:0]N);
      wire [63:0]B;
      wire [47:0]C;
      wire [31:0]D;
      wire [19:0]E;
      wire [11:0]F;
      genvar i;
      generate
       for (i=0;i<32;i=i+1) begin: Fa1bit
            assign #(1*37,1*36) {B[2*i+1],B[2*i]} = A[3*i+1] + A[3*i] + A[3*i+2];
       end
       for (i=0;i<16;i=i+1) begin: Fa2bit
            assign #(2*37,2*36) {C[3*i+2],C[3*i+1],C[3*i]} = {B[4*i+3],B[4*i+2]} +
{B[4*i+1],B[4*i]} + A[96+i];
       end
       for (i=0;i<8;i=i+1) begin: Fa3bit
            assign #(3*37,3*36) {D[4*i+3],D[4*i+2],D[4*i+1],D[4*i]} =
{C[6*i+5],C[6*i+4],C[6*i+3]} + {C[6*i+2],C[6*i+1],C[6*i]} + A[112+i];
       end
       for (i=0;i<4;i=i+1) begin: Fa4bit
            assign #(4*37,4*36) {E[5*i+4],E[5*i+3],E[5*i+2],E[5*i+1],E[5*i]} =
{D[8*i+7],D[8*i+6],D[8*i+5],D[8*i+4]} + {D[8*i+3],D[4*i+2],D[8*i+1],D[8*i]} + A[120+i];
       end
       for (i=0;i<2;i=i+1) begin: Fa5bit
            assign #(5*37,5*36) {F[6*i+5],F[6*i+4],F[6*i+3],F[6*i+2],F[6*i+1],F[6*i]} =
{E[10*i+9],E[10*i+8],E[10*i+7],E[10*i+6],E[10*i+5]} +
{E[10*i+4],E[10*i+3],E[10*i+2],E[10*i+1],E[10*i]} + A[124+i];
       end
       for (i=0;i<1;i=i+1) begin: Fa6bit
            assign #(6*37,6*36) N =
{F[12*i+11],F[12*i+10],F[12*i+9],F[12*i+8],F[12*i+7],F[12*i+6]} +
{F[12*i+5],F[12*i+4],F[12*i+3],F[12*i+2],F[12*i+1],F[12*i]} + A[126+i];
       end
      endgenerate
endmodule
```

Synthesized file of this part: Synthesized_P4.v

Part b. Synthesizing Problem 6's module:

For this part we most make Problem 6's module as a synthesizable description. So we make file as Verilog description (.v) .

File: ones_counter_P6_synable.v

```
`timescale 1ns/1ns
module ones_counter_synable(input [126:0]A,output reg [6:0]N);
      integer i;
      always @ (A) begin
       #777 N=7'd0;
       for (i=0;i<127;i=i+1) begin
            if (A[i] == 1'b1) N=N+1;
       end
      end
endmodule
```

```
Removed 0 unused modules.

2.23. Printing statistics.

=== ones_counter_synable ===

   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_synable..
Found and reported 0 problems.
```

```
4.1.2. Re-integrating ABC results.
ABC RESULTS:               NAND cells:     1290
ABC RESULTS:                NOR cells:     1779
ABC RESULTS:                NOT cells:      540
ABC RESULTS:        internal signals:     3187
ABC RESULTS:           input signals:      127
ABC RESULTS:          output signals:        7
Removing temp directory.
```

Synthesized file of this part: Synthesized_P6.v

Result:

We have more gates and cells in synthesized of problem 4 (with for and always).
