PROIECT

Aparate Electronice de Masurare si Control

Nume:Abunei Dumitrel

Profesor Indrumator : Andries Daniela

Anul : lll

Specializarea : MON

Prezentarea generală a temelor de proiect

1. Torsiometru digital –prezentare generală

Torsiometrul este un aparat destinat măsurării unghiului de torsiune la arborii în mișcare de rotație. Schema aplicației este prezentată în Figura 1:

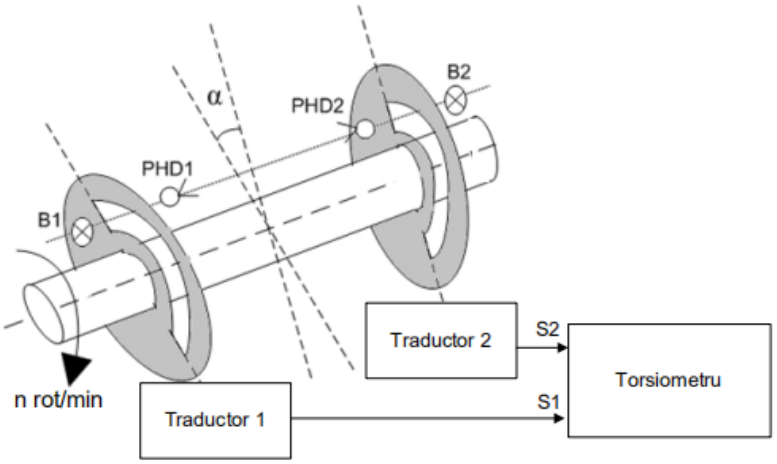


Figura 1 Schema de funcționare a torsiometrului

Schema bloc a torsiometrului digital, raportată la cerințele de implementare, este prezentată în Figura 2:

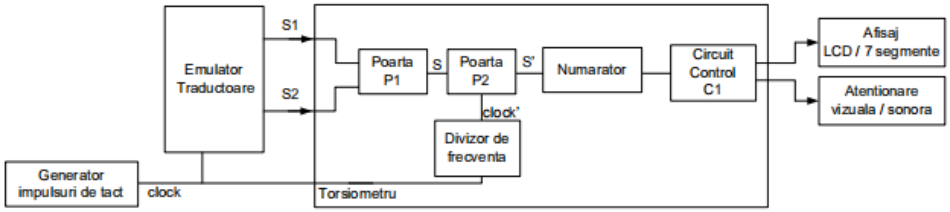


Figura 2 Diagrama bloc a aplicației

Blocul emulator traductoare permite generarea a diferite forme de undă provenite de la cele două traductoare, (S1, S2). Se generează forme de undă cu diferite valori a defazajului dintre ele, astfel încât să se acopere toate cazurile posibile de funcționare a aplicației. La ieșirea porții

P1 se obțin impulsuri de durată t, egală cu decalajul în timp al semnalelor aplicate la intrările torsiometrului. La intrarea clock' se aplică impulsurile obținute de la un traductor tahometric având un factor de multiplicare k față de frecvența semnalului dat de traductorul torsiometric :

$$k = \frac{T}{T'}$$

Poarta P2 permite trecerea impulsurilor de la iesirea circuitului divizor de frecvență numai în intervalul de timp t, ce reprezintă decalajul dintre semnalele primite de la traductoare. Numărul de impulsuri determinat și apoi afișat este :

$$N = \frac{t}{T'}$$

Indicatia N a aparatului este proporțională cu defazajul dintre semnale, respectiv unghiul de torsiune α.

Diagrama de impulsuri care descrie funcționarea aparatului este dată în Figura 3

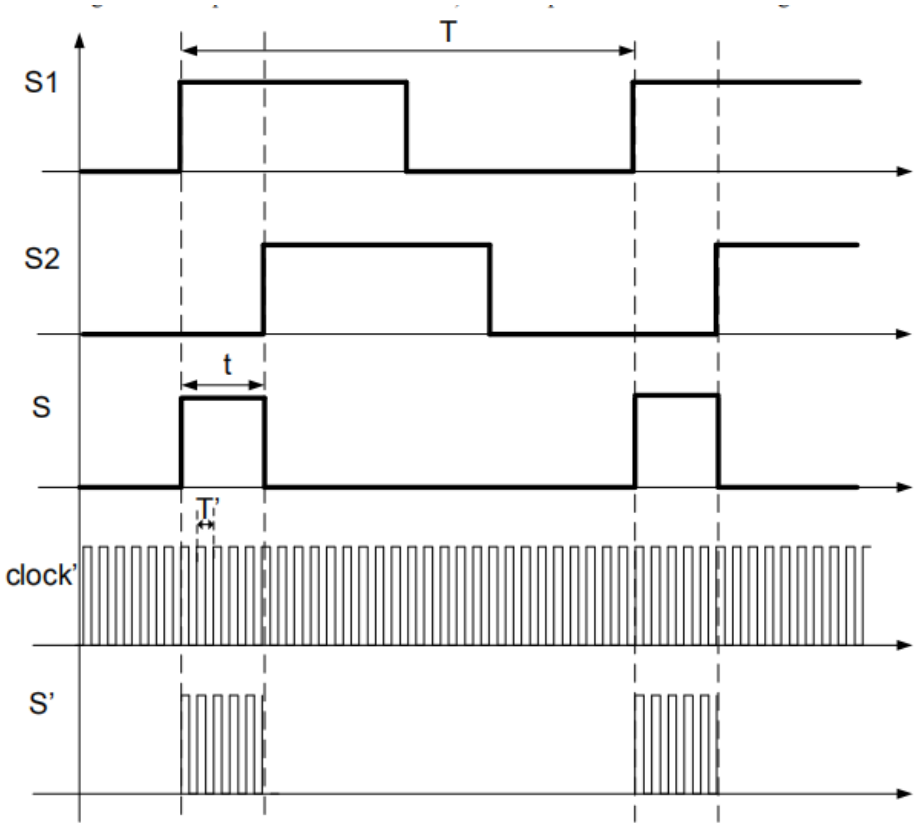
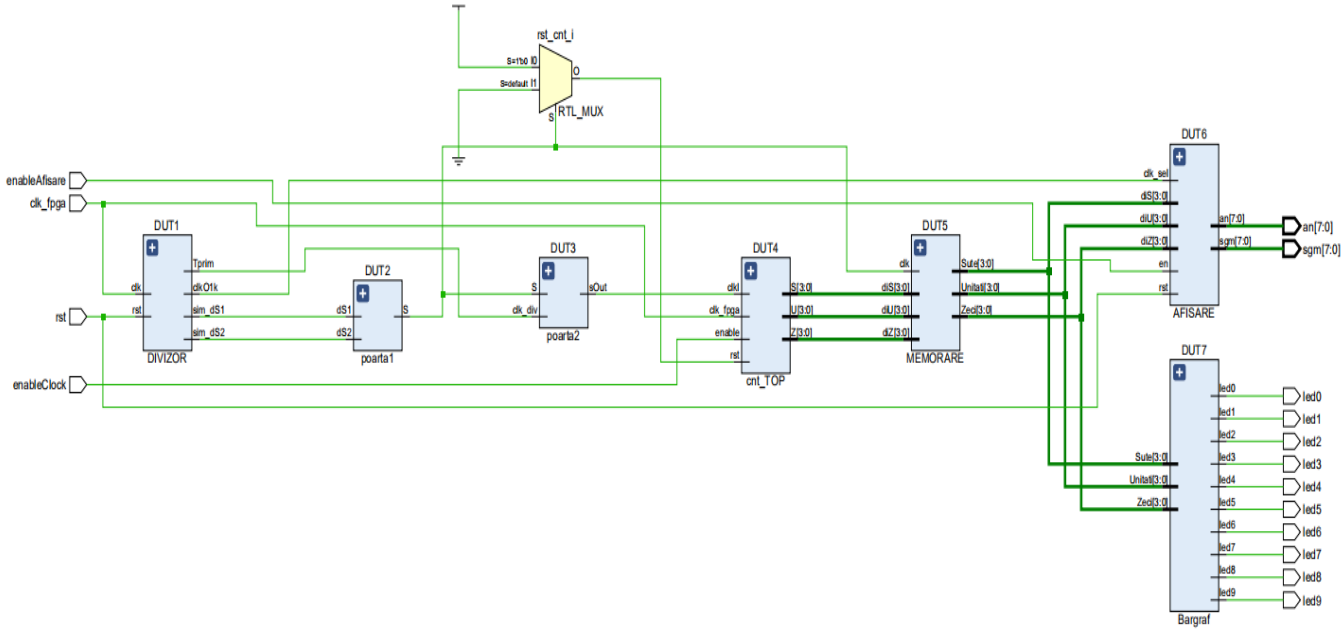
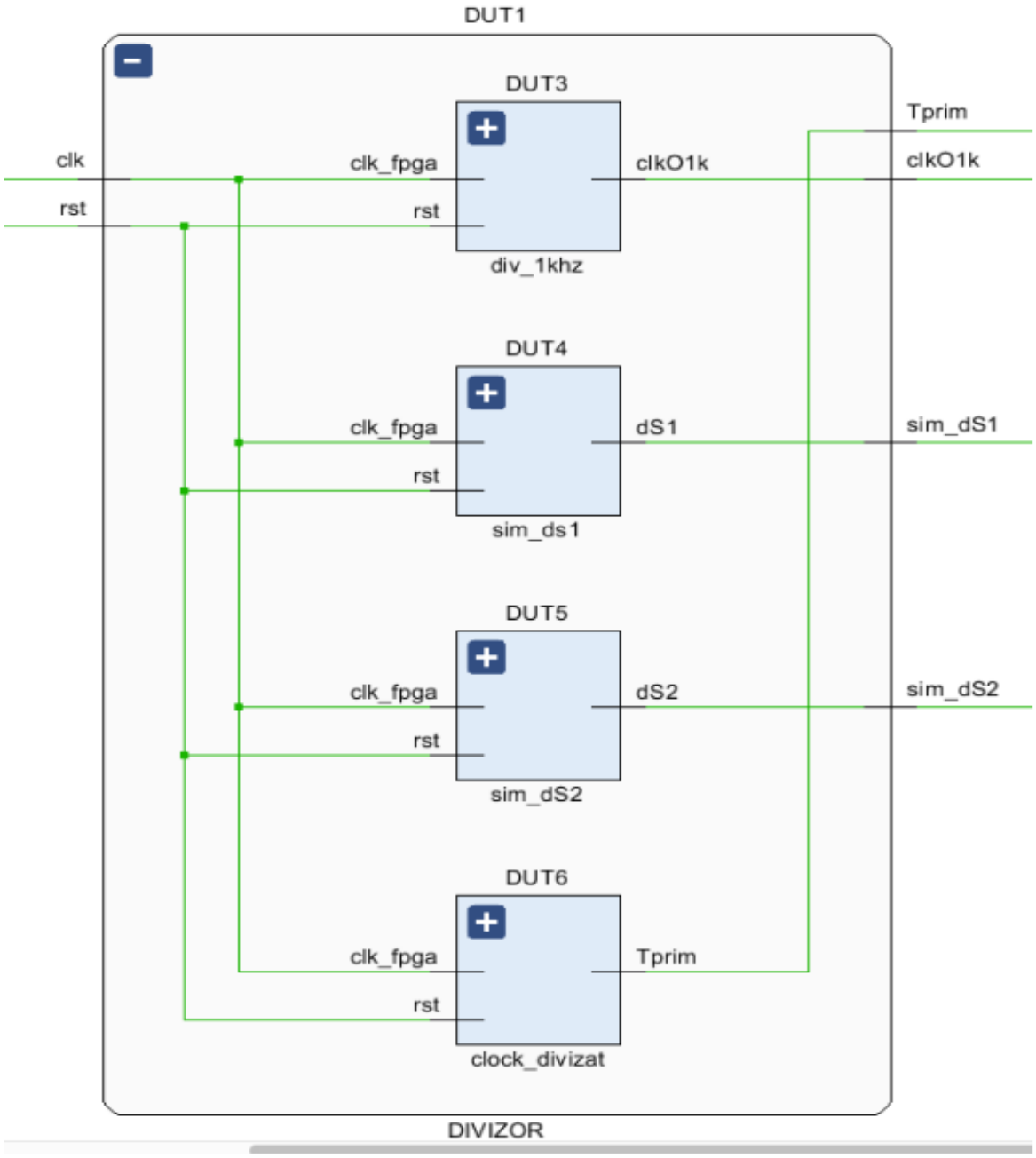


Figura 3 Diagrama de semnale


rst_cnt_i
S=1'b0 I0
S=default I1
O
S
RTL_MUX
enableAfisare
clk_fpga
rst
enableClock
DUT1
clk
rst
Tprim
clkOfk
sim_dS1
sim_dS2
DIVIZOR
DUT2
dS1
dS2
S
poarta1
DUT3
S
clk_div
sOut
poarta2
DUT4
clkI
clk_fpga
enable
rst
S[3:0]
U[3:0]
Z[3:0]
cnt_TOP
DUT5
clk
diS[3:0]
diU[3:0]
diZ[3:0]
Sute[3:0]
Unitati[3:0]
Zeci[3:0]
MEMORARE
DUT6
clk_sel
dS[3:0]
dU[3:0]
dZ[3:0]
en
rst
an[7:0]
sgm[7:0]
AFISARE
an[7:0]
sgm[7:0]
DUT7
Sute[3:0]
Unitati[3:0]
Zeci[3:0]
led0
led1
led2
led3
led4
led5
led6
led7
led8
led9
Bargraf
led0
led1
led2
led3
led4
led5
led6
led7
led8
led9

Implementarea divizoarelor de frecventa.



Module TOP divizor

```
module DIVIZOR(
   input clk,rst,
   output clkO1k,sim_dS1,sim_dS2,Tprim
   );

   div_1khz DUT3(
   .clk_fpga(clk),
   .rst(rst),
   .clkO1k(clkO1k)
   );

   sim_ds1 DUT4(
      .clk_fpga(clk),
      .rst(rst),
      .dS1(sim_dS1)
   );
   sim_dS2 DUT5(
      .clk_fpga(clk),
      .rst(rst),
      .dS2(sim_dS2)
   );
      clock_divizat DUT6(
      .clk_fpga(clk),
      .rst(rst),
      .Tprim(Tprim)
      );
   Endmodule

`timescale 1ns / 1ps
//  Frecventa semnalului clkO1k este de 1kHz
//  Se conecteaza mai departe la modulul de AFISARE
module div_1khz#(
```

```
    parameter MAX=49999
    )
    (
    input clk_fpga,rst,
    output reg clkO1k =0
    );

    reg [16:0]tmp=0;

    always@(posedge clk_fpga,posedge rst)
    begin
       if(rst)
       begin
          tmp<=0;
          clkO1k<=0;
        end
    else if(tmp==MAX)
       begin
          tmp<=0;
          clkO1k<=1;
        end
    else
       begin
        clkO1k<=0;
        tmp<=tmp+1;
        end
    end

endmodule
```

```
//  Frecventa semnalului Tprim este de 3.703MHz
//  Se conecteaza mai departe la Poarta 2
module clock_divizat#(
   parameter MAX=26     // N-1
)
(
   input clk_fpga,rst,
   output reg Tprim=0
   );
   reg [4:0]tmp=0;
   always@(posedge clk_fpga,posedge rst)
   begin
      if(rst)
      begin
         tmp<=0;
          Tprim<=0;
      end
   else if(tmp==MAX)
   begin
         tmp<=0;
         Tprim<=1;
   end
      else
      begin
         Tprim<=0;
         tmp<=tmp+1;
         end
    end
End module
```

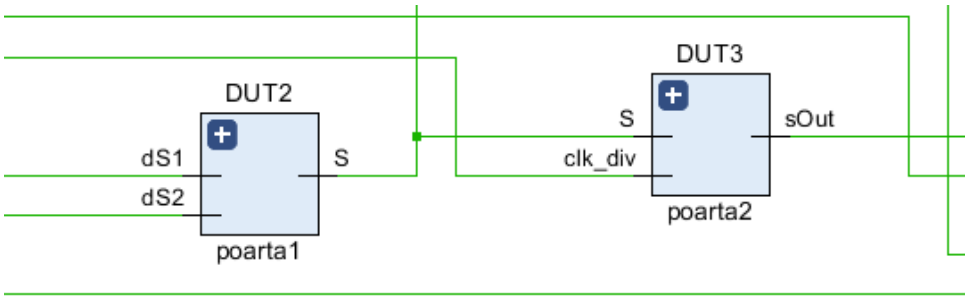
Implementarea formelor de unda.

```
module sim_ds1#(
   parameter MAX=9999999 // Factor de divizare N-1
)                    // Perioda semnalului aleasa este de 0.1s
( // Numarul de rotatii ales este 600 in intervalul de 60s
   input clk_fpga,rst,
   output reg dS1=0
   );
   reg [23:0]tmp=0;
   always@(posedge clk_fpga,posedge rst)
      begin
      if(rst)
         begin
         tmp<=0;
          dS1<=0;
         end
   else if(tmp==MAX)
       begin
       tmp<=0;
        dS1<=1;
       end
   else
   begin
      if(tmp<=MAX/2)
      begin
      dS1<=1;
      tmp<=tmp+1;
      end
      else if(tmp>=MAX/2)
```

```
      begin
       dS1<=0;
       tmp<=tmp+1;
      end
        else
   begin
   tmp<=tmp+1;
   end
   end
   end
Endmodule

module sim_dS2#(
   parameter MAX=9999999,
   parameter n=2699 // n reprezinta factorul de divizare al defazajului(N-1)
)              // unghi de torsiune ales = 1 grad
(   // Pt unghiul maxim 3 grade , valoarea n va fi n=8099(N-1)
   input clk_fpga,rst,
   output reg dS2=0
   );
   reg [23:0]tmp=0;
   reg [11:0]tmp_n=0; // se va modifica dim vectorului pt unghiul maxim [12:0]tmp_n

   always@(posedge clk_fpga,posedge rst)
   begin
      if(rst)
         begin
         tmp<=0;
         tmp_n<=0;
          dS2<=0;
         end
```

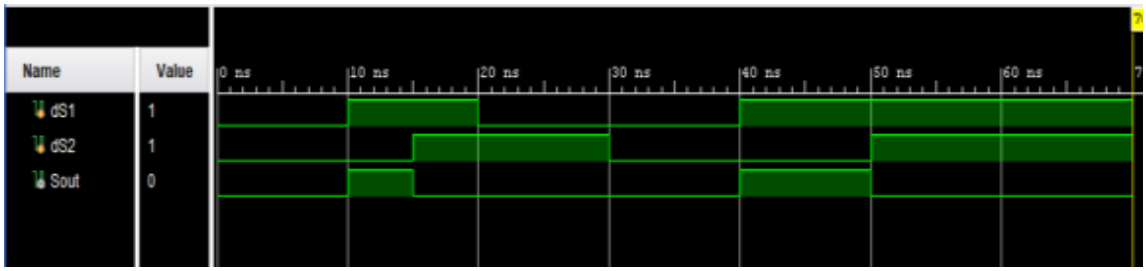
```
    else if(tmp_n==n)
    begin
    if(tmp==MAX)
        begin
        tmp<=0;
        dS2<=1;
        end
    else
    begin
       if(tmp<=MAX/2)
       begin
       dS2<=1;
       tmp<=tmp+1;
       end
       else if(tmp>=MAX/2)
       begin
        dS2<=0;
        tmp<=tmp+1;
       end
          else
    begin
    tmp<=tmp+1;
    end
    end
    end
       else
       begin
       tmp_n<=tmp_n+1;
       end
       end
Endmodule
```

Implementarea portilor logice.

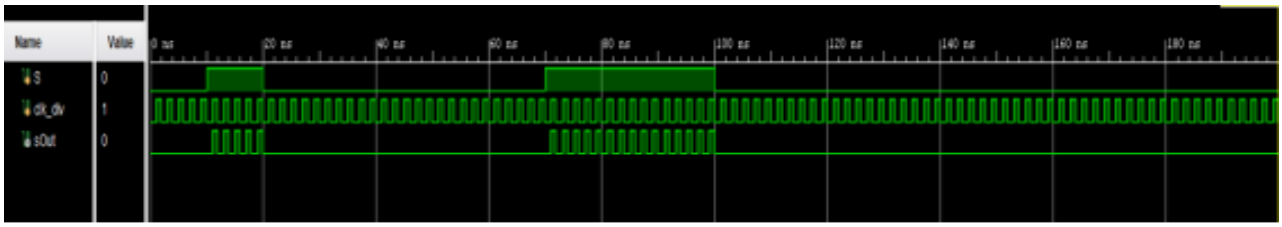



```
module poarta1(
   input dS1,dS2,
   output reg S = 0
   );
   always@*
   S = dS1&&(dS1^dS2);
  //  assign S = dS1&&(dS1^dS2);

Endmodule

module poarta2(
   input S,clk_div,
   output reg sOut
   );

   always@*
   if(S&&(clk_div||~clk_div))
      sOut<=clk_div;
      else
      sOut<=0;

endmodule
```

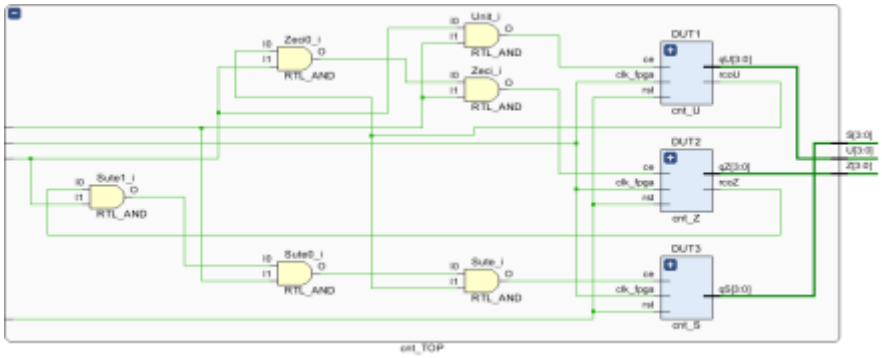
```
module tb_poarta1(
    );
    reg dS1,dS2;
    wire Sout;

    poarta1 DUT1(
            .dS1(dS1),
            .dS2(dS2),
            .S(Sout)
    );
    initial
    begin
    dS1=0;
    #10 dS1=1;
    #10 dS1=0;
    #20 dS1=1;
    end

    initial
    begin
     dS2=0;
     #15 dS2=1;
     #15 dS2=0;
     #20 dS2=1;
     End

     initial #70 $stop;
    Endmodule
```




```
module tb_poarta2(
    );
    reg S,clk_div;
    wire sOut;
    poarta2 DUT(
            .S(S),
            .clk_div(clk_div),
            .sOut(sOut)
            );
    initial
    begin
    S=0;
    clk_div=0;
    #10 S=1;
    #10 S=0;
    #50 S=1;
    #30 S=0;
    end
    initial #200 $stop;
    always #1 clk_div=~clk_div;
Endmodule
```



Implementarea Numaratorului




```
module cnt_TOP(
    input rst,clk_fpga,
    input clkI,enable,
    output [3:0]U,Z,S
    );
    wire tmpU,tmpZ,tmpS;
    wire Unit,Zeci,Sute;

    cnt_U DUT1(
        .ce(Unit),
        .rst(rst),
        .clk_fpga(clk_fpga),
        .qU(U),
        .rcoU(tmpU)
    );
```

```
  cnt_Z DUT2(
      .ce(Zeci),
      .rst(rst),
      .clk_fpga(clk_fpga),
      .qZ(Z),
      .rcoZ(tmpZ)
   );

   cnt_S DUT3(
      .ce(Sute),
      .rst(rst),
      .clk_fpga(clk_fpga),
      .qS(S)
   );

   assign Unit=enable&&clkI;
   assign Zeci=tmpU&&enable&&clkI;
   assign Sute=tmpZ&&enable&&clkI&&tmpU;

Endmodule
```

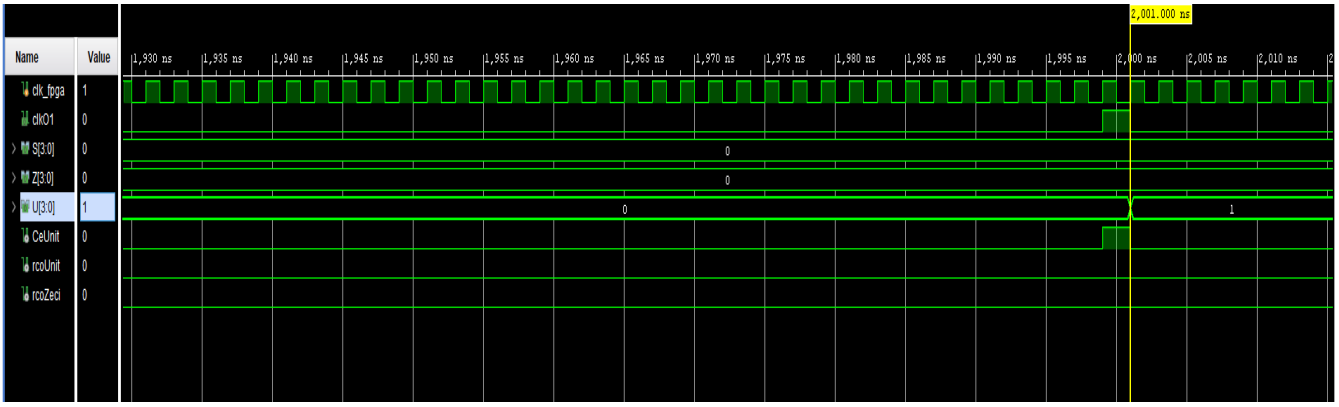
```
module cnt_U(
   input rst,clk_fpga,
   input ce,
   output rcoU,
   output reg [3:0]qU=0
   );

   always@(posedge clk_fpga,posedge rst)
   begin
   if(rst)
      qU<=0;
      else if(ce)
         if(qU==9)
         qU<=0;
   else
      qU<=qU+1;
   end

   assign rcoU=(qU==9)?1:0;

Endmodule
```

```
module cnt_Z(
   input rst,clk_fpga,
   input ce,
   output rcoZ,
   output reg [3:0]qZ=0
   );

   always@(posedge clk_fpga,posedge rst)
   begin
   if(rst)
      qZ<=0;
      else if(ce)
         if(qZ==9)
         qZ<=0;
   else
      qZ<=qZ+1;
   end

   assign rcoZ=(qZ==9)?1:0;

Endmodule
```

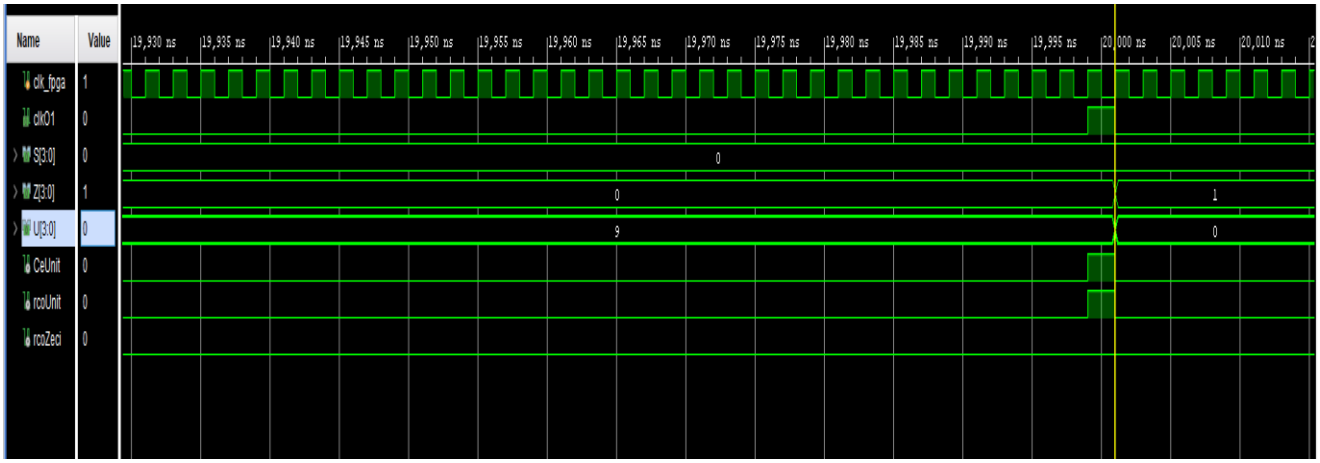
```
module cnt_S#(
   MAX=9
)(
   input rst,clk_fpga,
   input ce,
   output reg [3:0]qS=0
   );

   always@(posedge clk_fpga,posedge rst)
   begin
   if(rst)
      qS<=0;
   else if(ce)
         if(qS==9)
         qS<=0;
   else
      qS<=qS+1;
   end

Endmodule
```

Simularea Numaratorului.

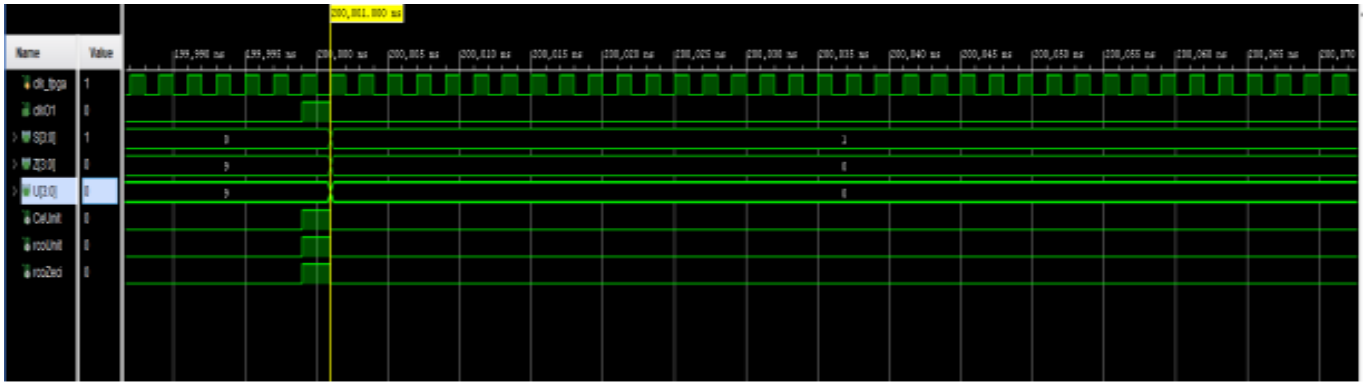
Counter at 1:



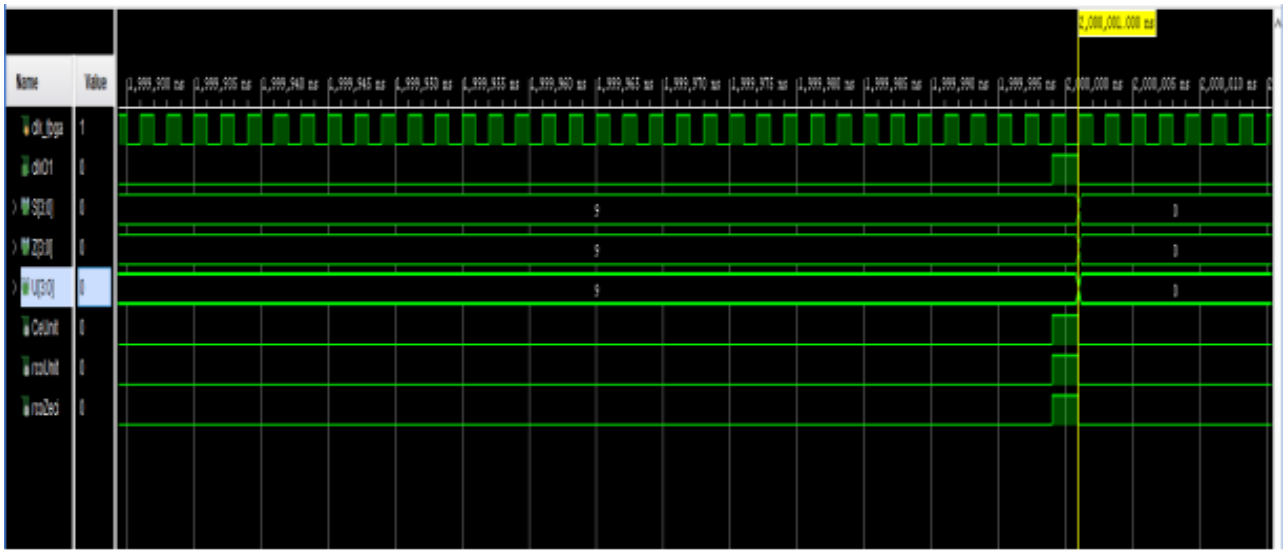
Counter at 10:



Counter at 100:

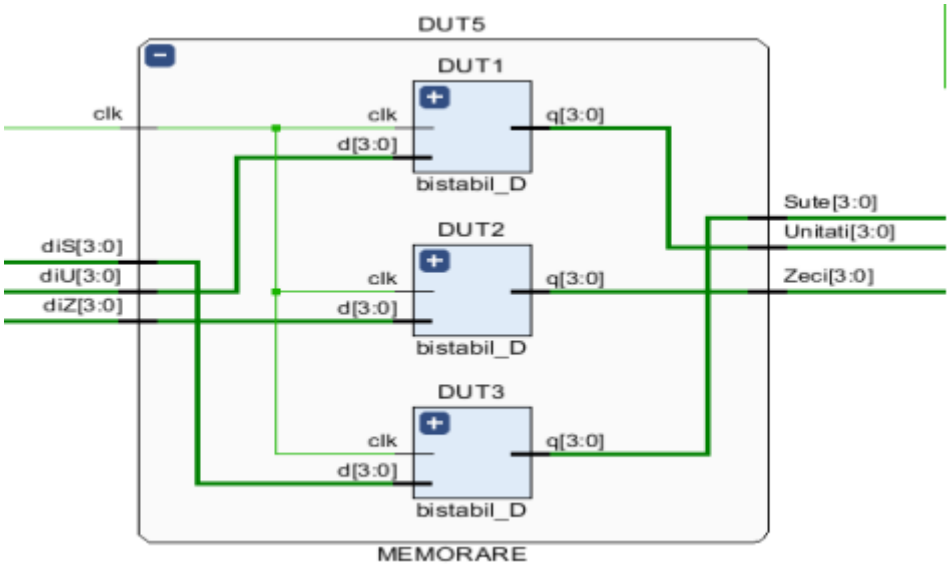


Counter at 999:



Implementarea circuitului de Memorare

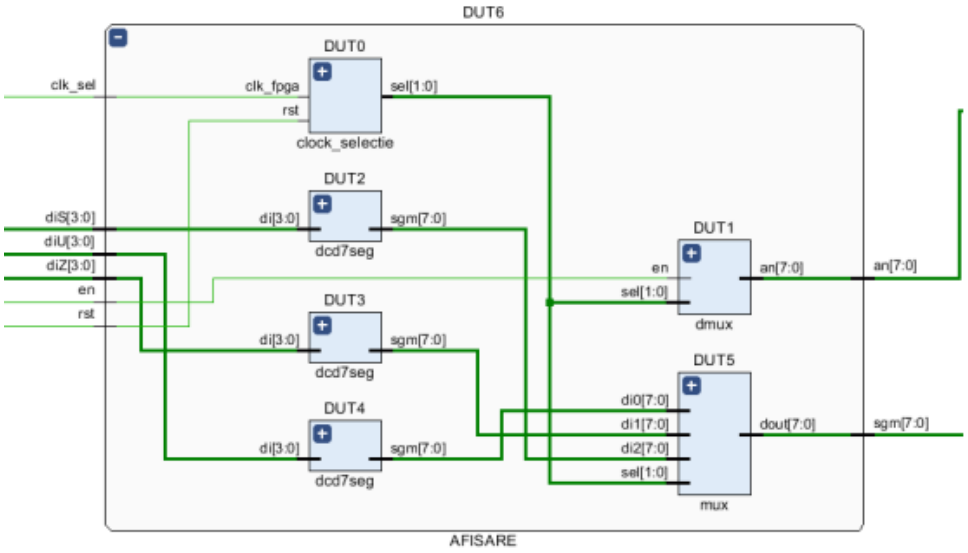
Circuitul de memorare este format din 3 bistabile si va memora ultima stare a numaratorului.



```
module MEMORARE(
    input clk,
    input [3:0]diU,diZ,diS,
    output [3:0]Unitati,Zeci,Sute
    );
    bistabil_D DUT1(
        .clk(clk),
        .d(diU),
        .q(Unitati)
    );
    bistabil_D DUT2(
        .clk(clk),
        .d(diZ),
        .q(Zeci)
    );
      bistabil_D DUT3(
        .clk(clk),
        .d(diS),
        .q(Sute)
    );
  Endmodule
```

```
`timescale 1ns / 1ps
// Pt circuitul de memorare s-a folosit semnalul de defazaj pe post de semnal de // clock
module bistabil_D(
    input clk,
    input [3:0]d,
    output reg [3:0]q=0
    );

always@(negedge clk)
    q<=d;
 Endmodule
```

Circuitul de afisare



Moudulul TOP pentru circuitul de afisare este urmatorul :

```
module AFISARE(
  input clk_sel,rst,en,
  input [3:0]diS,diZ,diU,
  output [7:0]an,sgm
   );
    wire [1:0]tmpCtrl;
   wire [7:0]tmpW2,tmpW1,tmpW0;

   clock_selectie DUT0(
      .clk_fpga(clk_sel),
      .rst(rst),
      .sel(tmpCtrl)
   );

   dmux DUT1(
      .sel(tmpCtrl),
      .en(en),
      .an(an)
   );

    dcd7seg DUT2(
      .di(diS),
      .sgm(tmpW2)
   );
```

```
dcd7seg DUT3(
        .di(diZ),
        .sgm(tmpW1)
    );

     dcd7seg DUT4(
        .di(diU),
        .sgm(tmpW0)
    );

        mux DUT5(
        .di0(tmpW0),
        .di1(tmpW1),
        .di2(tmpW2),
        .sel(tmpCtrl),
        .dout(sgm)
        );


endmodule
```

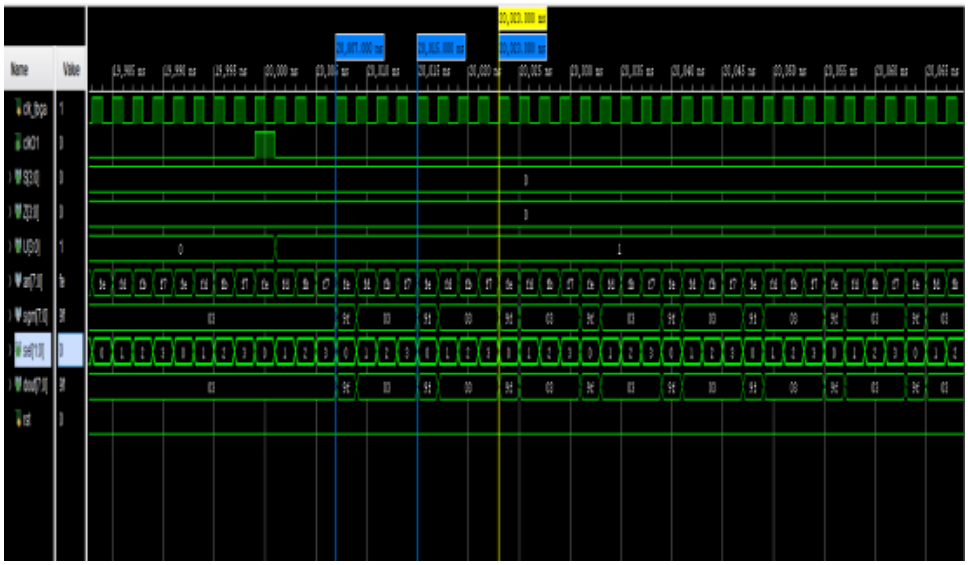
Submodule :

```
module clock_selectie(
   input clk_fpga,rst,
   output [1:0]sel
   );

 reg [1:0]refresh_cnt=0;

always@(posedge clk_fpga,posedge rst)
begin
   if(rst==1)
      refresh_cnt<=0;
   else
      refresh_cnt<=refresh_cnt+1;
End

   assign sel=refresh_cnt[1:0];

endmodule

  module dmux(
   input [1:0]sel,
   input en,
   output reg[7:0]an
   );

   always@(sel,en)
```

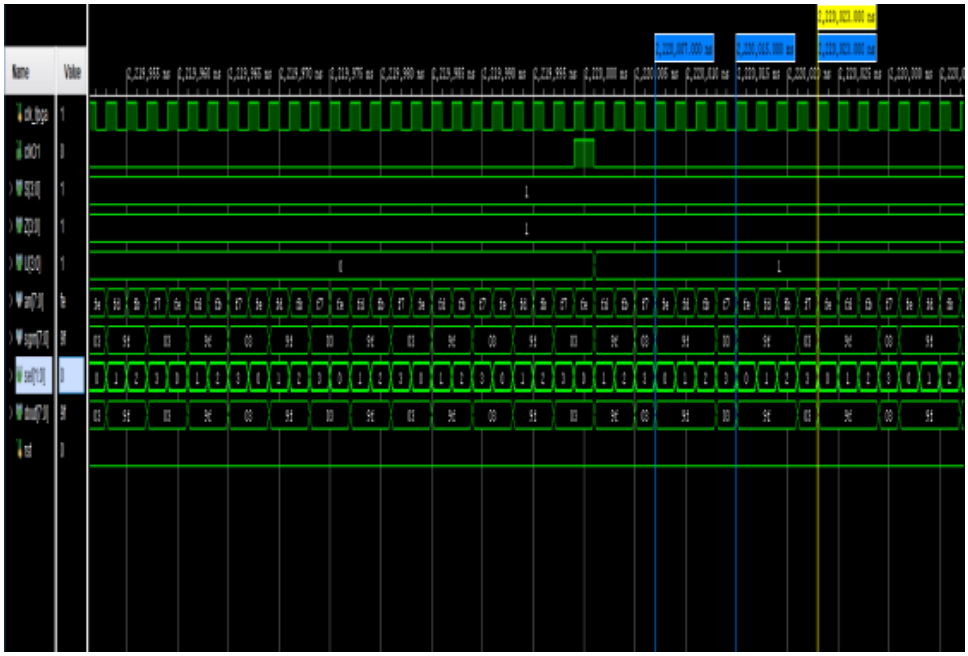
```
   case(sel)
   2'b0:begin an={1'b1,1'b1,1'b1,1'b1,1'b1,1'b1,1'b1,en}; end
   2'b1:begin an={1'b1,1'b1,1'b1,1'b1,1'b1,1'b1,en,1'b1}; end
   2'b10:begin an={1'b1,1'b1,1'b1,1'b1,1'b1,en,1'b1,1'b1}; end
   2'b11:begin an={1'b1,1'b1,1'b1,1'b1,en,1'b1,1'b1,1'b1}; end
   default:  an=8'b11111111;
   endcase
  Endmodule

module mux(
   input [7:0]di0,di1,di2,
   input [1:0]sel,
   output reg [7:0]dout
   );
   always@(sel,di0,di1,di2)
   begin
   case(sel)
   0:dout=di0;
   1:dout=di1;
   2:dout=di2;
   3:dout=8'b00000011;  // A 4-a celula de afisare va avea constant 0
   default:dout=8'bxxxx_xxxx; // Numaratorul implementat poate numara pana
   endcase   // la 999 dupa care incepe din nou de la 0.
   end

endmodule


module dcd7seg(
```

```
    input [3:0]di,
    output reg [7:0]sgm
    );

always@(di)
    case(di)
    0: sgm = 8'b00000011;
    1: sgm = 8'b10011111;
    2: sgm = 8'b00100101;
    3: sgm = 8'b00001101;
    4: sgm = 8'b10011001;
    5: sgm = 8'b01001001;
    6: sgm = 8'b01000001;
    7: sgm = 8'b00011111;
    8: sgm = 8'b00000001;
    9: sgm = 8'b00001001;
    10:sgm = 8'b11111111;
    11:sgm = 8'b11111111;
    12:sgm = 8'b11111111;
    13:sgm = 8'b11111111;
    14:sgm = 8'b11111111;
    15:sgm = 8'b11111111;
 default : sgm = 8'b11111111;
 endcase

 Endmodule
```

Afisare pt numarator la valoarea 1:



Afisare pentru numarator la valoarea 111:



Avertizare Bargraf:

```
module Bargraf(
   input [3:0]Unitati,Zeci,Sute,
   output led0,led1,led2,led3,led4,led5,led6,led7,led8,led9
   );

   assign led0=(Sute==0&&Zeci==0&&Unitati==0)?1:0;
   assign led1=(Sute==0&&Zeci<=2&&Unitati<=9)?1:0;
   assign led2=(Sute==0&&Zeci<=5&&Unitati<=9)?1:0;
   assign led3=(Sute==0&&Zeci<=9&&Unitati<=9)?1:0;
   assign led4=(Sute==1&&Zeci<=2&&Unitati<=9)?1:0;
   assign led5=(Sute==1&&Zeci<=5&&Unitati<=9)?1:0;
   assign led6=(Sute==1&&Zeci<=9&&Unitati<=9)?1:0;
   assign led7=(Sute==2&&Zeci<=2&&Unitati<=9)?1:0;
   assign led8=(Sute==2&&Zeci<=5&&Unitati<=9)?1:0;
   assign led9=(Sute==2&&Zeci<=9&&Unitati<=9)?1:0;
Endmodule
```

Modul TOP:

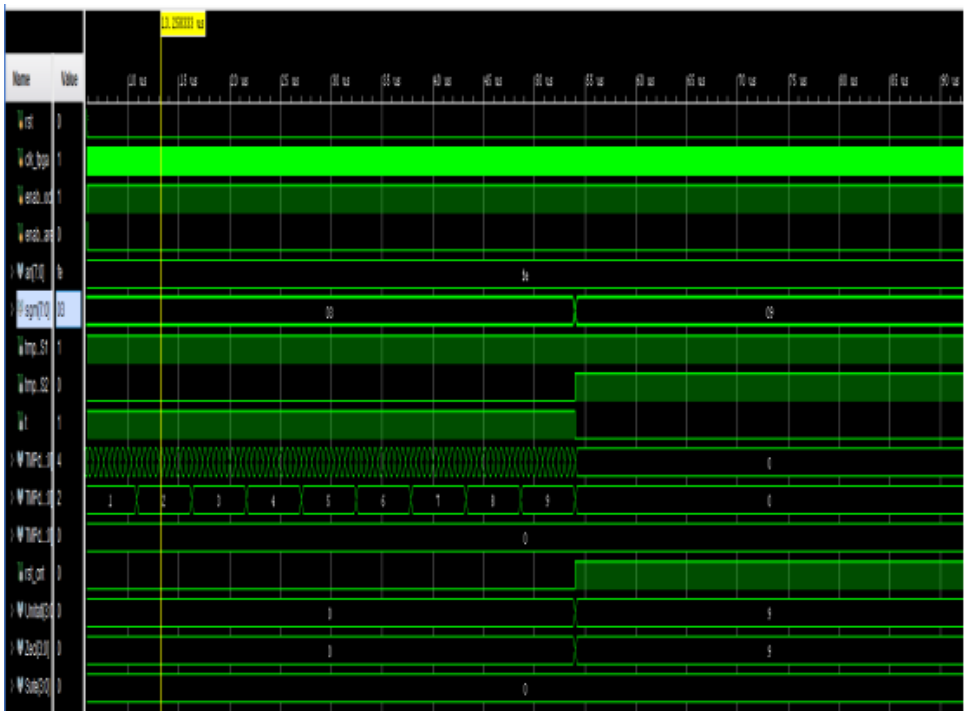
module TOP10(

```
 input rst,clk_fpga,enableClock,enableAfisare,
 output [7:0]an,sgm,
 output led0,led1,led2,led3,led4,led5,led6,led7,led8,led9
);
wire tmpClkO1k;
wire [3:0]TMPdiU,TMPdiZ,TMPdiS;
wire [3:0]Unitati,Zeci,Sute;
wire tmp_sim_dS1,tmp_sim_dS2;
wire t;
wire Tprim;
wire N;
wire rst_cnt;

DIVIZOR DUT1(
 .clk(clk_fpga),
 .rst(rst),
 .clkO1k(tmpClkO1k),
 .sim_dS1(tmp_sim_dS1),
 .sim_dS2(tmp_sim_dS2),
 .Tprim(Tprim)
);
```

```
poarta1 DUT2(
 .dS1(tmp_sim_dS1),
 .dS2(tmp_sim_dS2),
 .S(t)
 );
 poarta2 DUT3(
 .S(t),
 .clk_div(Tprim),
 .sOut(N)
 );
cnt_TOP DUT4(
.rst(rst_cnt),
.clk_fpga(clk_fpga),
.clkI(N),
.enable(enableClock), U(TMPdiU),
.Z(TMPdiZ),
.S(TMPdiS)
);
MEMORARE DUT5(
    .clk(t),
    .diU(TMPdiU),
    .diZ(TMPdiZ),
    .diS(TMPdiS),
    .Unitati(Unitati),
    .Zeci(Zeci),
    .Sute(Sute)
);
```

```
AFISARE DUT6(
    .clk_sel(tmpClkO1k),
    .rst(rst),
    .en(enableAfisare),
    .diS(Sute),
    .diZ(Zeci),
    .diU(Unitati),
    .an(an),
    .sgm(sgm)
);
Bargraf DUT7(
    .Unitati(Unitati),
    .Zeci(Zeci),
    .Sute(Sute),
    .led0(led0),
    .led1(led1),
    .led2(led2),
    .led3(led3),
    .led4(led4),
    .led5(led5),
    .led6(led6),
    .led7(led7),
    .led8(led8),
    .led9(led9)
);
assign rst_cnt=(t==0)?1:0;
Endmodule
```

Alpha considerat = 1 grad



Alpha considerat maxim = 3 grade