

Proiect HDL: Generator de semnal digital

Hardware Description Languages

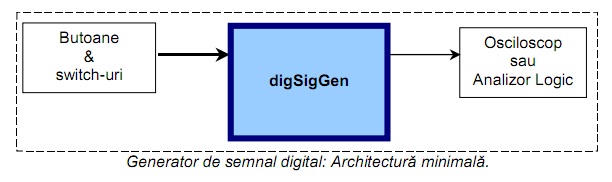
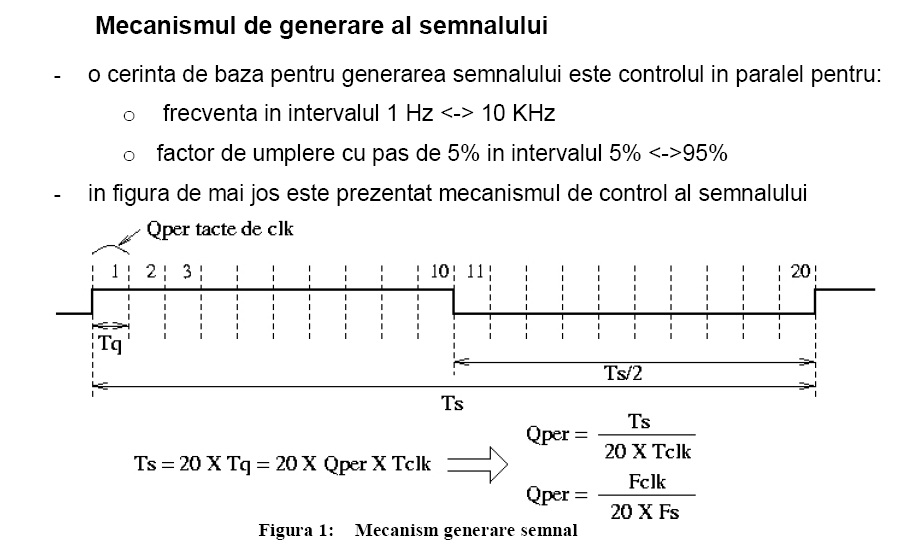
Anul de studiu: 3

Nume si prenume student: Virtej Florin Catalin

Grupa: 4981

Profesor : Gaspar Zoltan

Anul universitar: 2010-2011



**COD MODULE:**

1. **Selector\_frecv**

module selector\_frecv(sw1,sw2,sw3,sw4,clk,reset,pshbtn\_pl,pshbtn\_mi,frecv\_sel,q\_modif);

parameter width=16;

input sw1,sw2,sw3,sw4;

input clk;

input reset;

input pshbtn\_pl;

input pshbtn\_mi;

output [width-1:0] frecv\_sel;

output q\_modif;

reg [width-1:0] frecv\_reg;

reg [width-1:0] frecv\_sel;

reg q\_modif;

reg [width-1:0] comp1;

reg [width-1:0] comp2;

initial begin

frecv\_reg <= 1;

frecv\_sel <= 'b0;

q\_modif <= 0;

end

/\* always @ (posedge clk)

begin

if(frecv\_sel > 1)

comp1 <= 1;

else comp1 <= 0;

end

always @ (posedge clk)

begin

if(frecv\_sel < 10000)

comp2 <= 1;

else comp2 <= 0;

end \*/

always @ (posedge clk or reset)

begin

if(sw4&(pshbtn\_pl|pshbtn\_mi))

if(~pshbtn\_mi&pshbtn\_pl)

frecv\_sel <= frecv\_reg + 1000;

else frecv\_sel <= frecv\_reg - 1000;

if(sw3&(pshbtn\_pl|pshbtn\_mi))

if(~pshbtn\_mi&pshbtn\_pl)

frecv\_sel <= frecv\_reg + 100;

else frecv\_sel <= frecv\_reg - 100;

if(sw2&(pshbtn\_pl|pshbtn\_mi))

if(~pshbtn\_mi&pshbtn\_pl)

frecv\_sel <= frecv\_reg + 10;

else frecv\_sel <= frecv\_reg - 10;

if(sw1&(pshbtn\_pl|pshbtn\_mi))

if(~pshbtn\_mi&pshbtn\_pl)

frecv\_sel <= frecv\_reg + 1;

else frecv\_sel <= frecv\_reg - 1;

if(frecv\_sel > 1)

comp1 <= 1;

else comp1 <= 0;

if(frecv\_sel < 10000)

comp2 <= 1;

else comp2 <= 0;

if(comp2&comp1) begin

frecv\_reg <= frecv\_sel;

q\_modif <= 1'b1;

end

else begin

frecv\_reg <= frecv\_reg;

q\_modif <= 1'b0;

end

end

endmodule

1. **Duty\_cycle**

module duty\_cycle (pshbtn\_p, pshbtn\_m,clk,reset,t\_high,t\_low);

parameter width=5;

input pshbtn\_p;

input pshbtn\_m;

input clk;

input reset;

output [width-1:0] t\_high;

output [width-1:0] t\_low;

reg [width-1:0] data;

reg comp1,comp2;

initial begin

data <= 1010;

end

always @ (posedge clk)

begin

if(data > 1)

comp1 <= 1;

else comp1 <= 0;

end

always @ (posedge clk)

begin

if(data < 19)

comp2 <= 1;

else comp2 <= 0;

end

always @ (posedge clk or reset)

begin

if(reset)

data <= 1010;

else if((pshbtn\_p|pshbtn\_m)&(comp1&comp2)) begin

if (~pshbtn\_m & pshbtn\_p)

data <= data + 1; else

data <= data -1; end

else data <= data;

end

assign t\_high = data;

assign t\_low= 10100 - data;

endmodule

1. **divider**

module divider(clk,start,reset,ready,rest,cat,A,B );

parameter x = 24;

input clk;

input start;

input reset;

input [x-1:0] A;

input [x-1:0] B;

output ready;

output [x-1:0] rest;

output [x-1:0] cat;

wire reset\_dataW;

wire loadW;

wire readyCtrlW;

wire busyW;

cale\_de\_control #(x) cale\_de\_control(.start(start),.clk(clk),.reset(reset), .reset\_data(reset\_dataW) ,.load(loadW), .readyCtrl(readyCtrlW), .busy(busyW));

cale\_de\_date #(x) cale\_de\_date(.load(loadW),.reset(reset\_dataW),.busy(busyW), .clk(clk), .A(A), .B(B),.rest(rest),.cat(cat) ,.readyCtrl(readyCtrlW), .ready(ready));

endmodule

1. **cale\_de\_control**

module cale\_de\_control (start,reset,clk,reset\_data,load,readyCtrl,busy);

parameter width=24; //dimensiunea numaratorului in functie de timpul de lucru in care bussy =1

input start;

input reset;

input clk;

output reset\_data;

output load;

output readyCtrl;

output busy;

wire Q;

wire Q2;

wire Q3;

wire Q4;

wire Q5;

wire Q6;

bistabil bis(.D(start),.Q(Q),.clk(clk),.reset(reset));

bistabil bis2(.D(Q),.Q(Q2),.clk(clk),.reset(reset))

bistabil bis3(.D(Q2),.Q(Q3),.clk(clk),.reset(reset));

bistabil bis4(.D(Q3),.Q(Q4),.clk(clk),.reset(reset));

bistabil bis5(.D(Q4),.Q(Q5),.clk(clk),.reset(reset));

bistabil bis6(.D(Q5),.Q(Q6),.clk(clk),.reset(reset));

reg [width-1:0] numREG;

reg reset\_dataREG;

reg loadREG;

reg [2:0] readyCtrl1REG;

reg busyREG;

reg startREG;

reg readyREG;

initial begin

numREG <={width{1'b0}};

reset\_dataREG <= 0;

loadREG <= 0;

readyCtrl1REG <= 0;

busyREG <= 0;

startREG <= 0;

readyREG <=0;

end

always @(posedge clk)

begin

if((Q || Q2 || Q3 || Q4 || Q5) && ~busyREG)

begin

startREG <= 1;

if(startREG)

begin

readyREG <= 0 ;

readyCtrl1REG <= 3'b0;

end

end

else

startREG <= 0;

if(reset)

begin

numREG <={width{1'b0}};

reset\_dataREG <= 0;

loadREG <= 0;

readyCtrl1REG <= 0;

busyREG <= 0;

startREG <= 0;

readyREG <=0;

end

else

begin

if(startREG)

begin

if(Q2 == 1)

begin

reset\_dataREG <=1;

readyREG <=0;

readyCtrl1REG <= 3'b0;

end

else

if(Q3 == 1)

reset\_dataREG <=0;

else

if(Q4 == 1)

begin

loadREG <= 1;

end

else

if(Q5 == 1)

loadREG <=0;

else

if(Q6 == 1)

begin

busyREG <=1;

end

else

begin

reset\_dataREG <=0;

loadREG <=0;

end

end

if(busyREG)

numREG <= numREG + 1;

if(numREG == 72-2) // 24=8\*3 unde 8 e numarul de biti pe care il au operanzii impartiri. Durata

//impartirii este de 3 ori numarul de biti.

begin

numREG <= 0;

busyREG <=0;

readyREG <= 1;

end

end

readyCtrl1REG <= {readyREG,readyCtrl1REG[2:1]};

end

assign reset\_data = reset\_dataREG;

assign load = loadREG;

assign readyCtrl = readyCtrl1REG[0];

assign busy = busyREG;

endmodule

1. **cale\_de\_date**

module cale\_de\_date (load, reset, busy, clk, A, B, rest, cat, readyCtrl, ready );

parameter width=24;

input [width-1 :0] A;

input [width-1:0] B;

output [width-1 :0] rest;

output [width-1 :0] cat;

output ready;

input readyCtrl;

input load;

input clk;

input reset;

input busy;

reg [width:0] P;

reg [width-1:0] Areg;

reg [width-1:0] Breg;

reg [width-1:0] restReg;

reg [width-1:0] catReg;

reg readyReg;

reg [2:0] wait2clk;

reg msb;

reg [width:0]BregNeg;

initial begin

P <= {width{1'b0}};

Areg <= {width{1'b0}};

Breg <= {width{1'b0}};

restReg <= {width{1'b0}};

catReg <= {width{1'b0}};

readyReg <=0;

wait2clk <=3'b0;

msb <=0;

BregNeg <= {width{1'b0}};

end

always @(posedge clk or reset)

begin

BregNeg <=~Breg+1;

if (wait2clk == 2'b10)

wait2clk <= 2'b00;

else

wait2clk <= wait2clk+1;

if (reset)

begin

P <= {width{1'b0}};

Areg <= {width{1'b0}};

Breg <= {width{1'b0}};

restReg <= {width{1'b0}};

catReg <= {width{1'b0}};

readyReg <=1'b0;

wait2clk <=3'b00;

msb <=0;

BregNeg <= {width{1'b0}};

end

else

if (load)

begin

Areg <= A;

Breg <= B;

end

else

if (busy)

begin

if(wait2clk == 0)

begin

msb <= P[width];

P <= {P[width-1:0],Areg[width-1]};

end

if(wait2clk == 1)

begin

if(msb == 1)

P <= P+Breg;

if(msb == 0)

P <= P+BregNeg;

end

if(wait2clk == 2)

Areg <= {Areg[width-2:0],~P[width]};

end

if (readyCtrl)

begin

if(P[width] == 1)

restReg <=P[width-1:0]+B;

if(P[width] == 0)

restReg <= P[width-2:0];

catReg <= Areg[width-1:0];

readyReg <= 1;

end

end

assign rest = restReg;

assign cat = catReg;

assign ready = readyReg;

endmodule

1. **debouncer**

module debouncer (data,clk,reset,out);

parameter width=64;

input [width-1:0] data ;

input clk;

input reset;

output out;

reg [width-1:0] data;

reg out;

initial begin

data <= 'b0;

end

always @ (posedge clk or reset)

begin

if(reset)

data <= 'b0;

else if(&data)

out <=1;

else out <= 0;

end

endmodule

1. **bistabil**

module bistabil(clk,D,Q,reset );

input D;

input clk;

input reset;

output Q;

reg mem;

initial

begin

mem <=0;

end

always @(posedge clk)

begin

if (reset)

mem <=0;

else

mem <=D;

end

assign Q = mem;

endmodule

1. **constr\_frecv (generatorul de frecv)**

module constr\_frecv(t\_high,t\_low,clk,reset,signal\_o,pulse\_o);

parameter width=6;

input [width-1:0] t\_high;

input [width-1:0] t\_low;

input clk;

input reset;

input pulse\_o;

output [width-1:0] signal\_o;

reg [width-1:0] count1;

reg [width-1:0] count2;

reg load;

reg cen1,cen2;

reg carry1,carry2;

reg [width-1:0] signal\_o;

initial begin

end

always @ (posedge clk)

begin

if(count1==0)

carry1 <=1; else carry1 <= 0;

end

always @ (posedge clk)

begin

if(count2==0)

carry2 <=1; else carry2 <= 0;

end

always @ (posedge clk or reset)

begin

if (reset)

load <=1;

if (load) begin

count1 <= t\_high;

count2 <=t\_low;

end

else if(pulse\_o&(~carry1))

cen1 <= 1;

else if (cen1) begin

count1 <= count1 - 1;

signal\_o <= 1;

end

if (pulse\_o&(~carry2)&(~&count1))

cen2 <= 1;

else if(cen2) begin

count2 <= count2 -1;

signal\_o <= 0;

end

end

endmodule

1. **cuanta**

module cuanta (cat,q\_modif,reset,clk,pulse\_o);

parameter width=23;

input [width-1:0] cat;

input q\_modif;

input reset;

input clk;

output pulse\_o;

reg [width-1:0] count;

reg pulse\_o;

reg [width-1:0] cat\_hold;

always @ (posedge clk or reset)

begin

if(reset) begin

count <= 'b1;

cat\_hold <= 'b0; end

else if(pulse\_o)

if(q\_modif)

begin

cat\_hold <= cat;

count <= cat; end

else

count <= cat\_hold;

if(~&count)

pulse\_o <=1;

else pulse\_o <= 0;

end

endmodule