Universitatea
Transilvania
din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR

TEMA Limbaje de descriere hardware

Student: Gheorghe Lucian Giani Andrei
Grupa: 4LF291
Specializare: Electronica Aplicata

Circuit de impartire secvential

Circuit de impartire secvential:

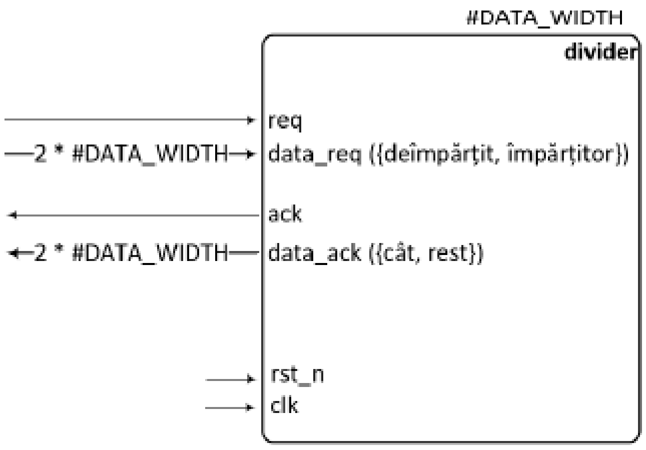
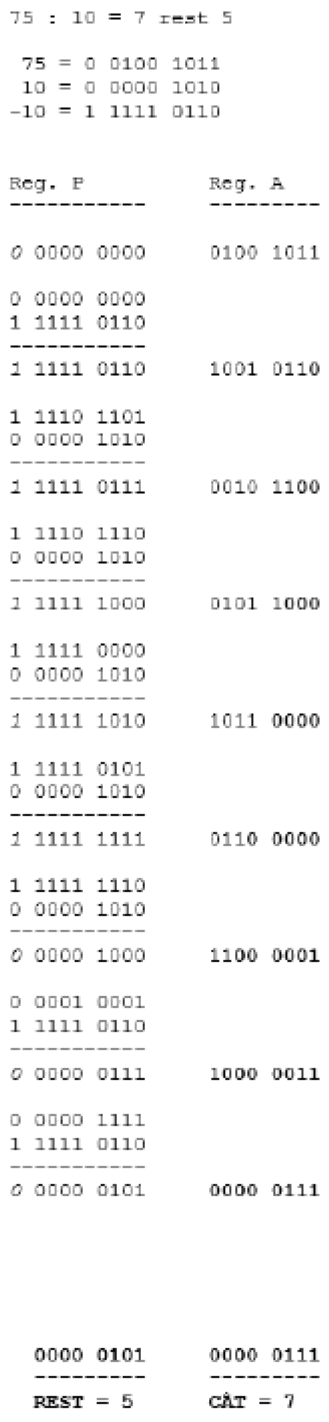


FIG.1 Circuit de impartire secvential

Algoritm de impartire fara restaurare, numere pozitive:

```
-reseteaza P (n+1 biti)
-incarca deimpartitul in A (n biti)
-incarca impartitorul in B (n biti)
-repeta de n ori
        -daca P este negativ (MSB = 1) atunci
                -deplaseaza cu o pozitie stanga P (LSB P = MSB A)
                -P <= P + B
        -altfel
                -deplaseaza cu o pozitie stanga P (LSB P = MSB A)
                -P <= P + (-B)
        -deplaseaza cu o pozitie in stanga A (LSB A = not MSB P)
-daca P este negativ (MSB = 1) atunci
        -P <= P + B
-P contine REST
-A contine CAT
```



```
75 : 10 = 7 rest 5

 75 = 0 0100 1011
 10 = 0 0000 1010
-10 = 1 1111 0110


Reg. P          Reg. A
-----------     ---------

0 0000 0000     0100 1011

0 0000 0000
1 1111 0110
-----------
1 1111 0110     1001 0110

1 1110 1101
0 0000 1010
-----------
1 1111 0111     0010 1100

1 1110 1110
0 0000 1010
-----------
1 1111 1000     0101 1000

1 1111 0000
0 0000 1010
-----------
1 1111 1010     1011 0000

1 1111 0101
0 0000 1010
-----------
1 1111 1111     0110 0000

1 1111 1110
0 0000 1010
-----------
0 0000 1000     1100 0001

0 0001 0001
1 1111 0110
-----------
0 0000 0111     1000 0011

0 0000 1111
1 1111 0110
-----------
0 0000 0101     0000 0111




  0000 0101     0000 0111
  ---------     ---------
  REST = 5      CÂT = 7
```

FIG.2 Parcurgerea impartiri pas cu pas

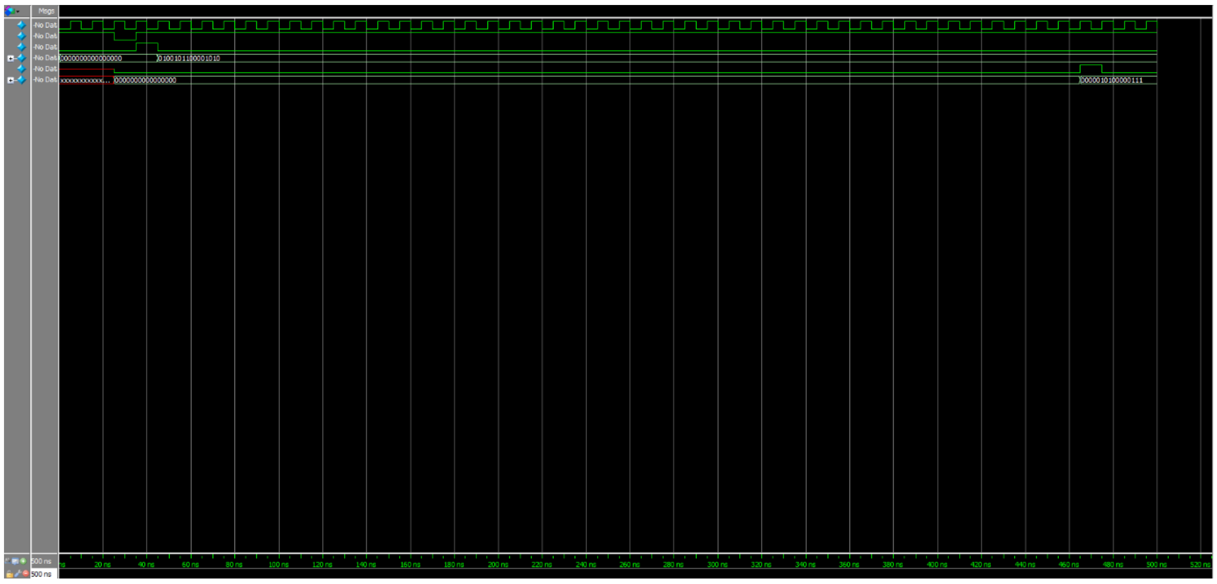


FIG.3 top waveform

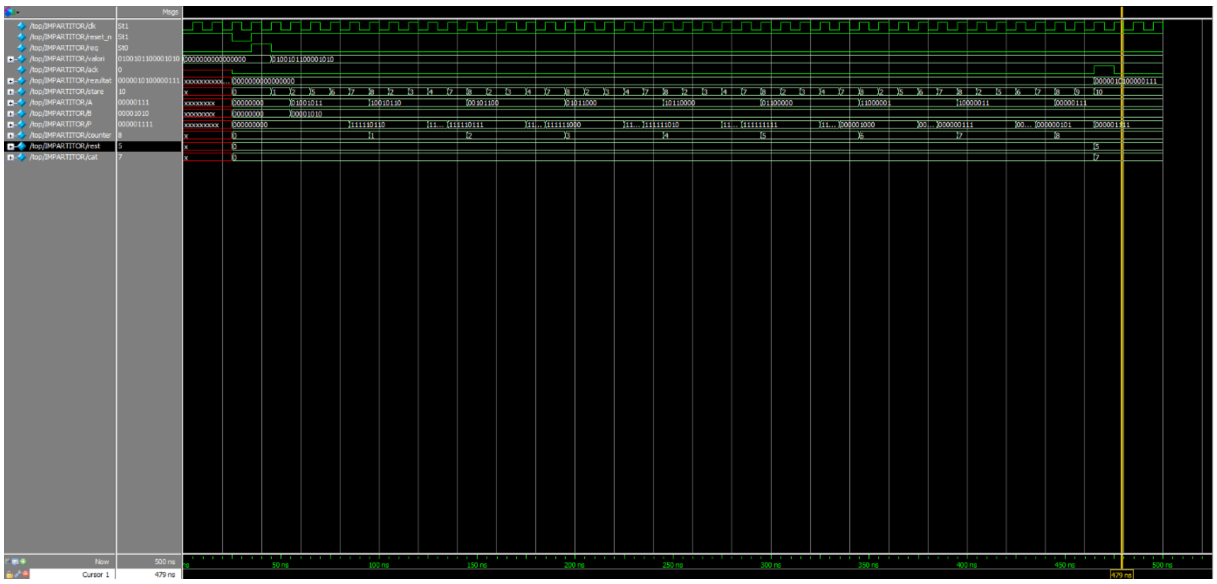


Fig.4 divider waveform

Cod verilog:

DUT:

```
//-----------------------------------------------------------------------------------
// Universitatea Transilvania din Brasov
// Proiect     : Limbaje de descriere hardware
// Autor       : <Gheorghe Lucian Giani Andrei>
// Data        : <29/05/2022>
//-----------------------------------------------------------------------------------
// Descriere   : <Testbench>
//-----------------------------------------------------------------------------------
module top();

reg              clk;
reg              reset_n;
reg              req;
reg  [15:0] valori;

wire             ack;
wire [15:0] rezultat;

always #5 clk <= ~clk;

initial begin
clk = 0;
reset_n = 1;
req = 0;
valori = 0;
repeat (3) @(posedge clk);
reset_n = 0;
@(posedge clk);
req = 1;
reset_n = 1;
@(posedge clk)
valori = 16'b0100101100001010;
req = 0;
@(posedge clk);
repeat (150) @(posedge clk);
$stop;
end

divider DIVIDER(
.clk(clk),
.reset_n(reset_n),
.req(req),
.valori(valori),
.ack(ack),
.rezultat(rezultat)
);

endmodule
```

DIVIDER:

```
//-----------------------------------------------------------------------------------
// Universitatea Transilvania din Brasov
// Proiect     : Limbaje de descriere hardware
// Autor       : <Gheorghe Lucian Giani Andrei>
// Data        : <29/05/2022>
//-----------------------------------------------------------------------------------
// Descriere   : <Divider>
//-----------------------------------------------------------------------------------
module divider(
input              clk,
input              reset_n,
input              req,
input  [15:0] valori,
output reg    ack,
output [15:0] rezultat
);

reg [3:0] stare;
reg [7:0] A;
reg [7:0] B;
reg [8:0] P;
reg [3:0] counter;
reg [7:0] rest;
reg [7:0] cat;

assign rezultat [15:8] = rest;
assign rezultat [7:0] = cat;

localparam S0  = 4'b0000; //reset_n
localparam S1  = 4'b0001; //incarcare deimpartitor si impartitor
localparam S2  = 4'b0010; //if msb p = 1
localparam S3  = 4'b0011; //concatenare (P) (LSB P = MSB A)
localparam S4  = 4'b0100; //P <= P + B
localparam S5  = 4'b0101; //concatenare (P) (LSB P = MSB A)
localparam S6  = 4'b0110; //P <= P + (-B)
localparam S7  = 4'b0111; //concatenare (A) (LSB A = not MSB P)
localparam S8  = 4'b1000; //if msb p = 1
localparam S9  = 4'b1001; //P <= P + B
localparam S10 = 4'b1010; //STOP
```

```
always @(posedge clk or negedge reset_n)
begin
        if (~reset_n) stare <= S0;
        else if (stare == S0 & req) stare <= S1;
        else if (stare == S1) stare <= S2;
        else if (stare == S2 & P[8]) stare <= S3;
        else if (stare == S3) stare <= S4;
        else if (stare == S2 & ~P[8]) stare <= S5;
        else if (stare == S5) stare <= S6;
        else if (stare == S4) stare <= S7;
        else if (stare == S6) stare <= S7;
        else if (stare == S7) stare <= S8;
        else if (stare == S8 & counter == 8) stare <= S9;
        else if (stare == S8) stare <= S2;
        else if (stare == S9) stare <= S10;
end

always @(posedge clk or negedge reset_n) //negedge
begin
        if (~reset_n) A <= 'b0;
        else if (stare == S1) A <= valori [15:8];
        else if (stare == S7) A [7:0] <= {A[6:0], ~P[8]};
end

always @(posedge clk or negedge reset_n)
begin
        if (~reset_n) B <= 'b0;
        else if (stare == S1) B <= valori [7:0];
end

always @(posedge clk or negedge reset_n) //negedge
begin
        if (~reset_n) P <= 'b0;
        else if (stare == S3 | stare == S5) P <= {P[7:0], A[7]};
        else if (stare == S4 | stare == S9) P <= P + B;
        else if (stare == S6) P <= P + (-B);
end

always @(posedge clk or negedge reset_n)
begin
        if (~reset_n) ack <= 'b0;
        else if (ack) ack <= 'b0;
        else if (stare == S9) ack <= 'b1;
        else if (stare == S10) ack <= 'b0;
end

always @(posedge clk or negedge reset_n)
```

```
begin
        if (~reset_n) rest <= 'b0;
        else if (stare == S9) rest <= P [7:0];
end

always @(posedge clk or negedge reset_n)
begin
        if (~reset_n) cat <= 'b0;
        else if (stare == S9) cat <= A;
end

always @(posedge clk or negedge reset_n)
begin
        if (~reset_n) counter <= 'b0;
        else if (stare == S0) counter <= 'b0;
        else if (stare == S7) counter <= counter + 1;
end

endmodule
```